
MLH localhost
Workshop
Blockchain Casino
Learning how to
SCORE on ICON
MLH localhost
ICON
MLH
MAJOR LEAGUE HACKING
v.0.0.1

Our Mission is to Empower Hackers.

65,000+
HACKERS

12,000+
PROJECTS CREATED

400+
CITIES

We hope you learn something awesome today!
Find more resources: http://mlh.io/

Table of Contents

What will you learn today?

1. An Introduction to Blockchain technology and ICON
2. How to create a Blockchain wallet to send and receive transactions
3. How to deploy an application to the ICON network using Morpheus Labs

When you hear the word ‘Blockchain’, what do you think about?

What is Blockchain?

To help us understand what Blockchain is, let's rewind to the days before the Internet and computers.

If you had a bank account, the details of all of your transactions would be stored in the bank ledger.

Key Term

Ledger: A store of financial records, in paper or digital form.

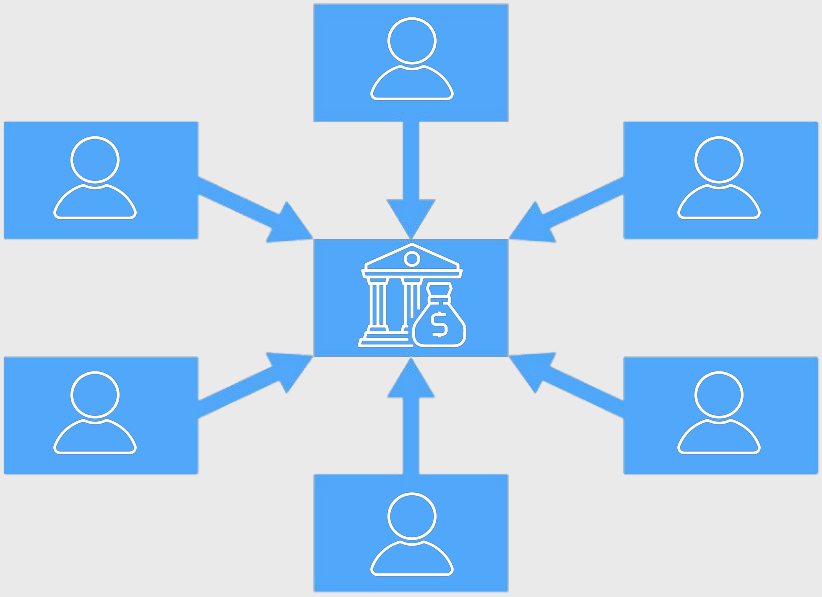
This is known as **centralized** banking.

All processes happen via the bank, and are recorded by the bank.

Centralized Banking

Think of the bank as being the center of all financial operations.

If you'd like to pay someone, you give the money to the bank, who then transfer it to the recipient.



Can you see any problems with Centralized banking?

Centralized Banking

Let's think about our paper bank ledger.

- What if someone tampers with it? It's the only source of truth for our bank records!
- What if the bank gets into major financial trouble?

Blockchain as a Solution

Blockchain was created to help solve the problems that can arise from having a single, centralised source of knowledge.

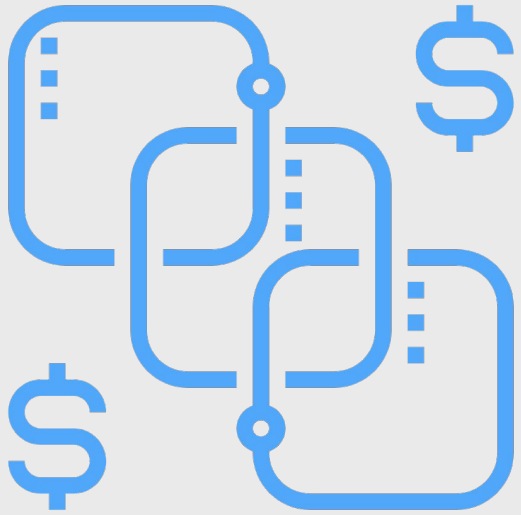
It sought to create a **decentralized** platform: one where there was no single source of truth, and all knowledge was shared **collectively**.

Blockchain as a Solution

Again, let's go back to our paper ledger.

What if....

- Everyone had a copy of the ledger.
- All new transactions were verified by everyone's ledger copy.
- Transactions were sent directly to one another (**peer to peer**), instead of through a central financial institution.



This is the foundation of Blockchain!

A History of Blockchain

The key to Blockchain is **cryptography**. Each new piece of information in the ledger is linked to the previous one with a highly secure **cryptographic hash**, making them incredibly difficult to modify.

Blockchain is *a chain of securely connected blocks of data*.

Key Term
Cryptographic Hash: An algorithm designed to encrypt data to keep it protected and secure.

Blockchain technology has been around since the 1990s, but really took off in the 2008 when an unknown person used Blockchain technology to create the cryptocurrency **Bitcoin.**

What can we use Blockchain for?

Mention Blockchain to people and they tend to start thinking about digital currency. This is one use for Blockchain, but not the only one!

Contracts: Blockchain can protect and preserve details of legal documents.

Healthcare: Private health records can be encrypted and securely transferred.

Music: Blockchain can create a decentralized database of ownership rights and royalty payments.

Table of Contents

Introducing ICON

The ICON project uses Blockchain to ensure the integrity of data being shared between organisations.

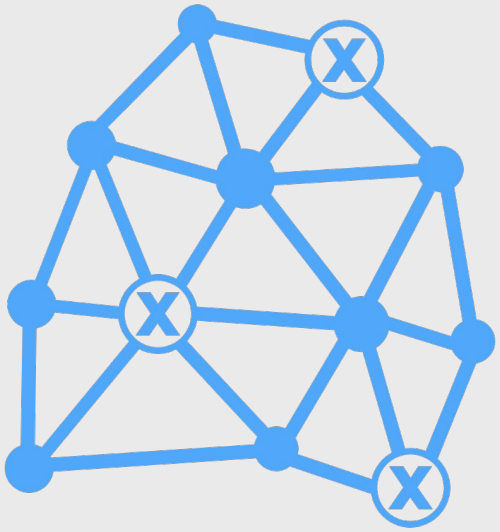
"The ICON Project aims to build a decentralized network that allows independent blockchains with different governances to transact with one another without intermediaries."

- The ICON Project white paper, 2018

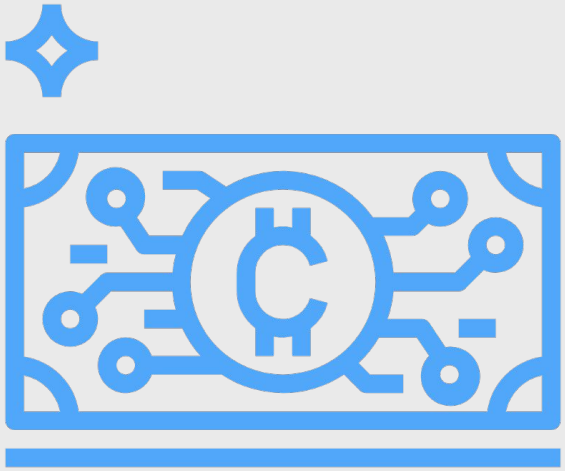
Introducing ICON

There are many different implementations of Blockchain technology, and it's difficult for them to communicate with one another.

ICON aims to allow these technologies to interact and transact with one another, without lots of third parties in the middle. This is what **decentralized governance** means.



Introducing ICON: ICX



Part of the ICON project is a cryptocurrency called ICON Exchange Tokens (ICX). We can give and receive ICX online.

We store our ICX in a digital **wallet**, a secure location for our ICX data.

Introducing ICON: Smart Contracts

A **SCORE** is a **S**mart **C**ontract **o**n a **R**eliable **E**nvironment.

A SCORE is deployed onto the ICON network - a block on the Blockchain.

We can send transactions to a SCORE's address, and all nodes on the ICON network verify the result of the transaction.

For example, if we send coins to a friend, we write a transaction which reduces our balance and simultaneously increases theirs.

T-Bears

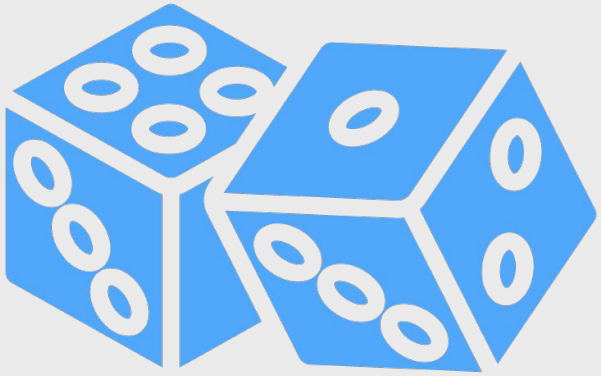
We can write SCOREs using **T-Bears**, ICON's suite of command-line development tools.

We can use T-Bears to create SCOREs and test them on ICON's TestNet - an environment that emulates the real life Mainnet of ICON.

Key Term

Emulator: Hardware or software which imitates the functions of another system.

What will you learn today?



We're going to use ICON technology to deploy a Slot Machine application where users bet ICX to play the game, and will either gain or lose ICX based on the outcome.

Let's see how it will look!

Provable Fairness

A hugely important aspect of online gaming is **trust**.

As a player, how can you know that the Slot Machine is truly random and fair, and that it hasn't been tampered with?

This concept is called **Provable Fairness**, and Blockchain can help to provide this trust to the user. Let's explain how!

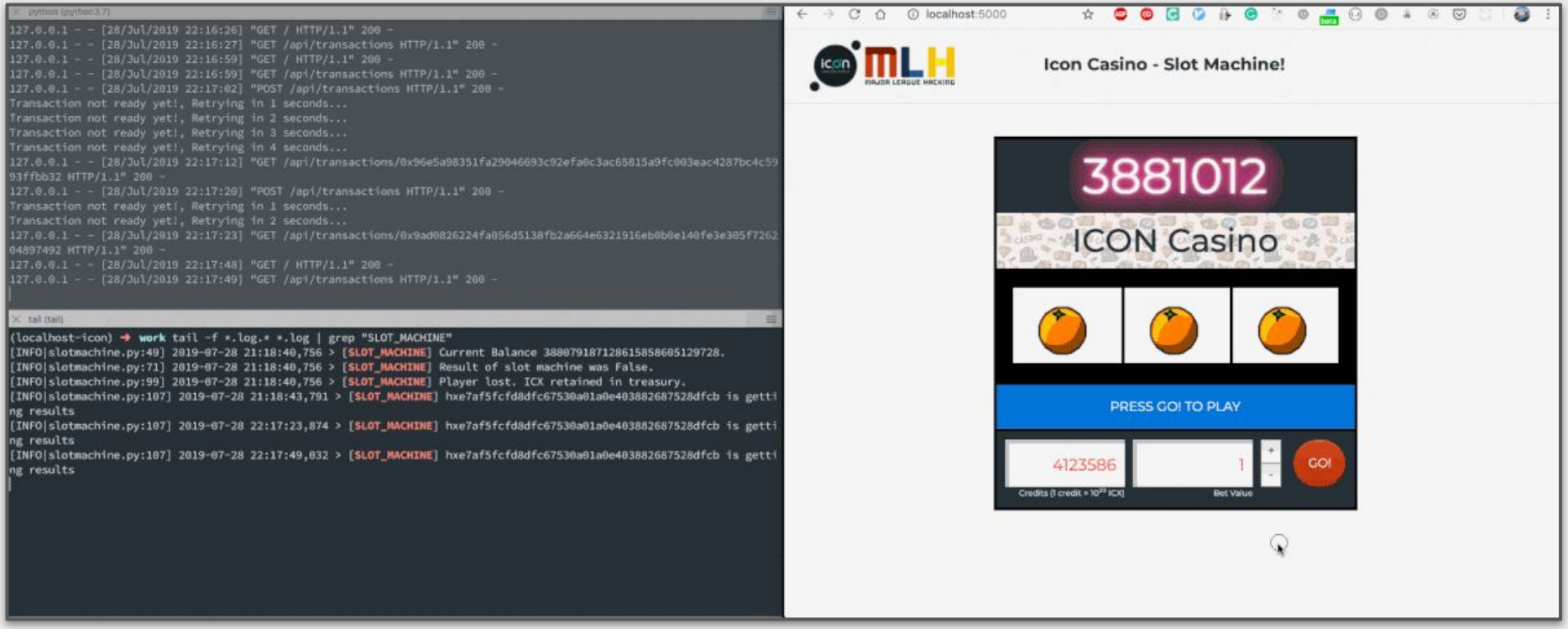
Provable Fairness

The gaming logic of our Slot Machine app will generate a random number. If the number ends in 0, the player has won.

To generate the number, our SCORE will combine a complex, verifiable transaction hash with the current timestamp and convert it into a number.

This way, our generated number can be trusted to be fair.

What will you learn today?



What will you learn today?

To create the application we will work through:

- Launching the Slot Machine application
- Setting up an ICX wallet
- Creating a Morpheus Labs workspace
- Deploying a Smart Contract (SCORE) onto the ICON TestNet

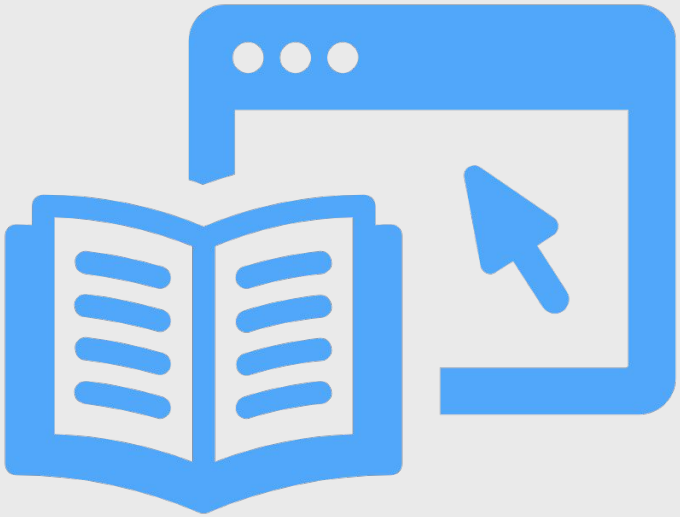
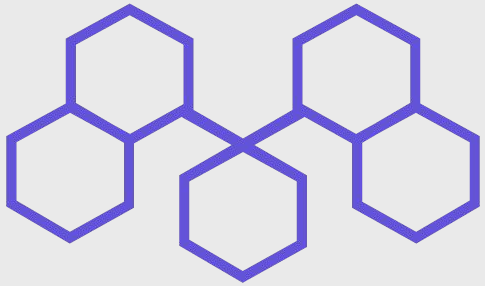


Table of Contents

Introducing Morpheus Labs



Morpheus Labs

Morpheus Labs is a Blockchain-Platform-as-a-Service (BPaaS).

It's a web-based tool that's going to allow us to write SCOREs, deploy them, and launch our web app - all from your browser.

Let's get started!

Creating a Morpheus Labs Account

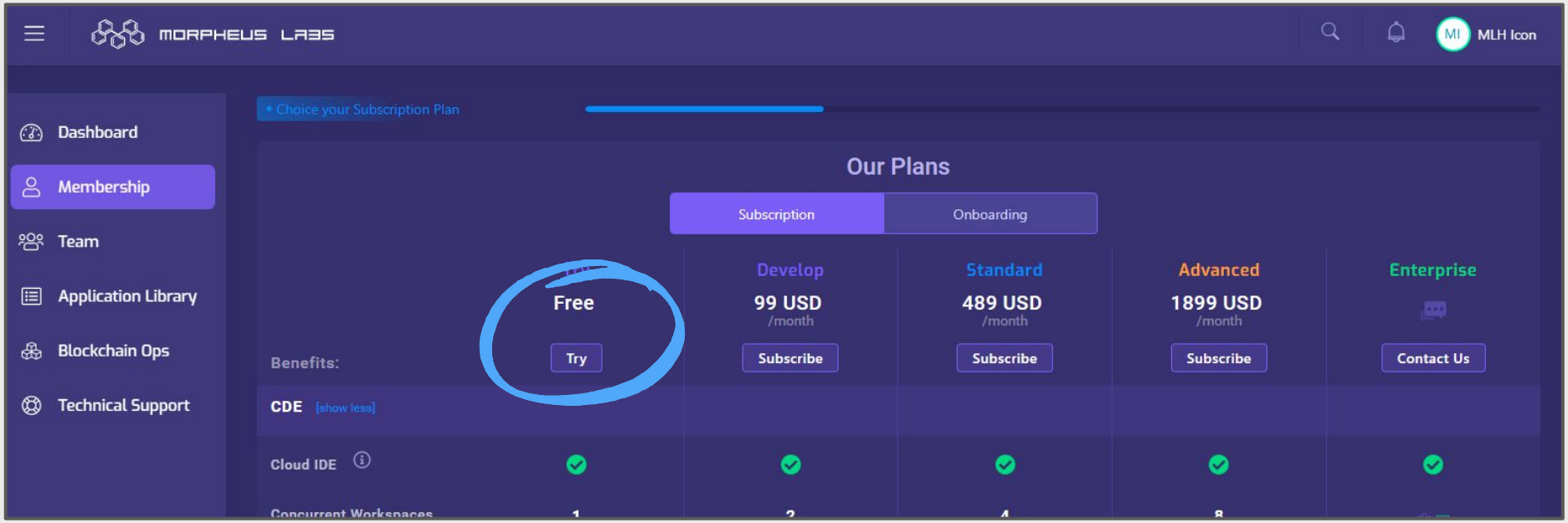
First off, we need to create an account with Morpheus Labs.

Head to this URL and click **Register New Account**:

mlhlocal.host/morpheus-login

Configuring Morpheus Labs

Click the **Try** plan, click **Checkout**, then **Confirm**.



Configuring Morpheus Labs

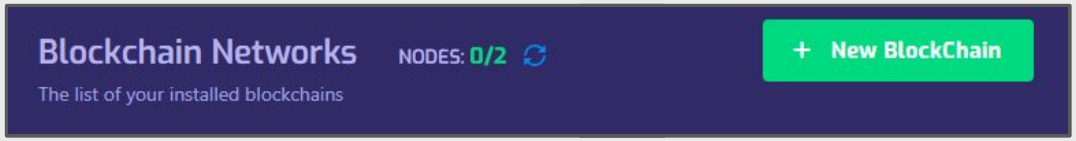
Our to-do list is:

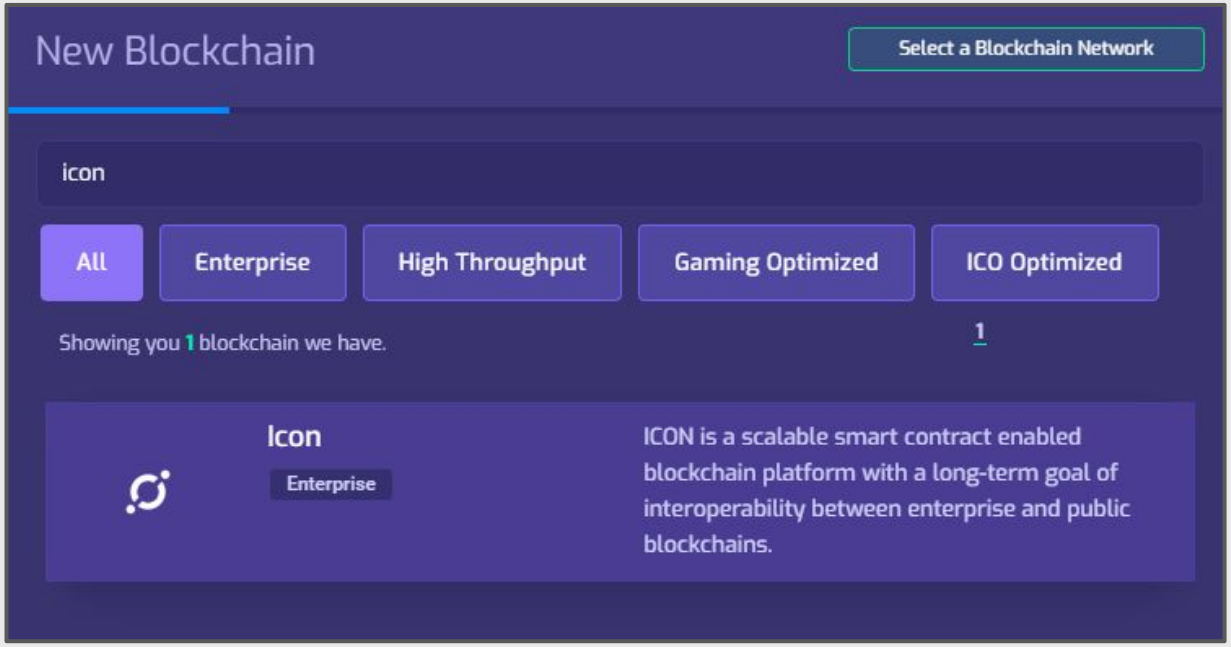
- Create a new ICON Blockchain Network
- Get the source code of the application from GitHub
- Create a new Workspace

Start by clicking the first button, **New Blockchain Network**.

Creating a Blockchain Network

First, select **New Blockchain**.

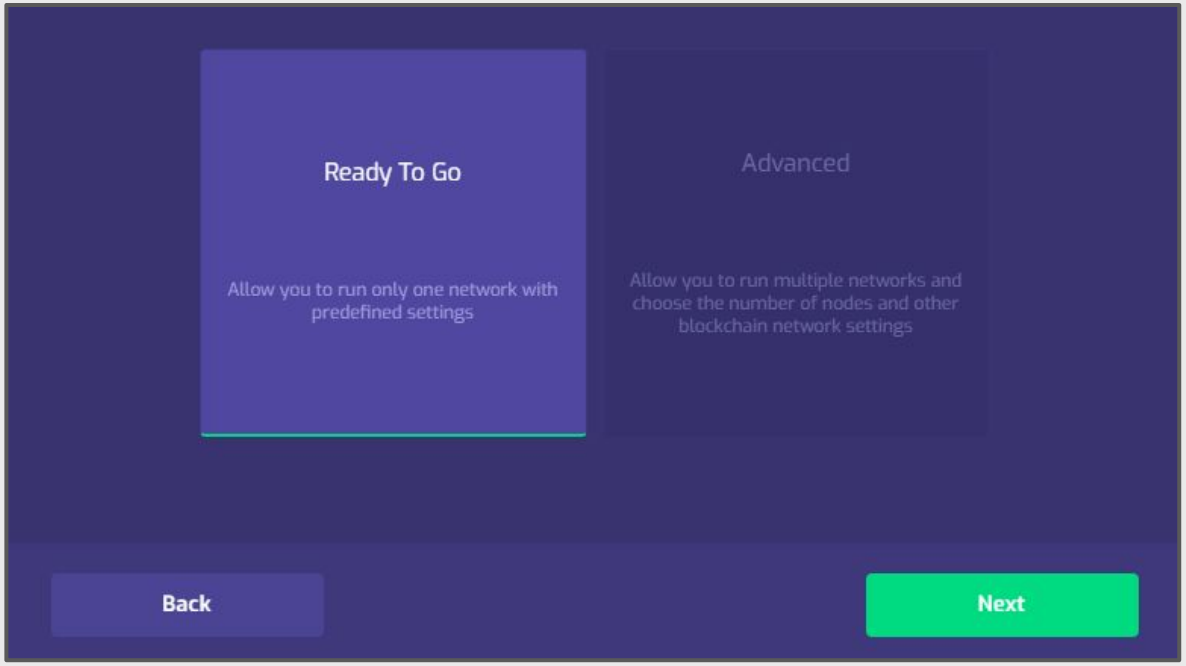




Next, search for 'icon' and select the **Icon** network.

Then click **Next**.

Creating a Blockchain Network

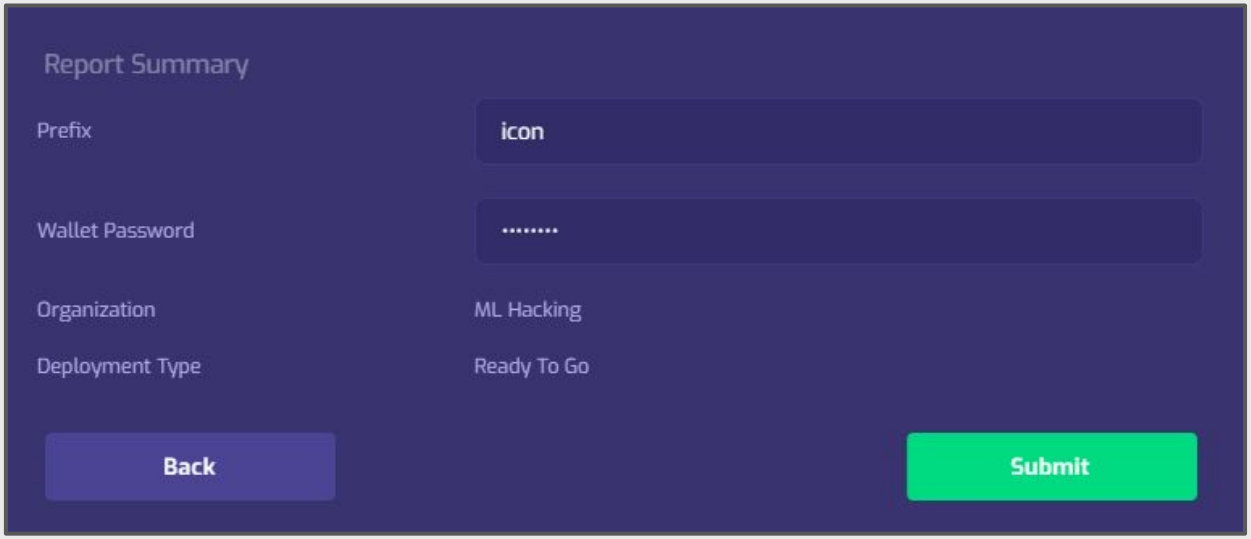


Keep the default **Ready to Go** option and click **Next**.

Creating a Blockchain Network

For the prefix type **icon**, and for the password, enter **password** for now.

Lastly, hit **Submit**!



Creating a Blockchain Network

Click the > button and the network will launch.

Let’s recap!

We’ve set up Morpheus Labs and created an ICON network.

Next, we need to get the source code for the web application.

Setting up GitHub

GitHub is a web platform to store and share code in repositories.

We're going to create a GitHub account and **fork** (copy) the starter code for this project.

To get started, head to this URL:

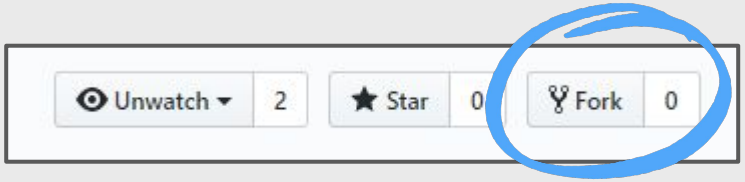
mlhlocal.host/github-signup

Getting the Source Code

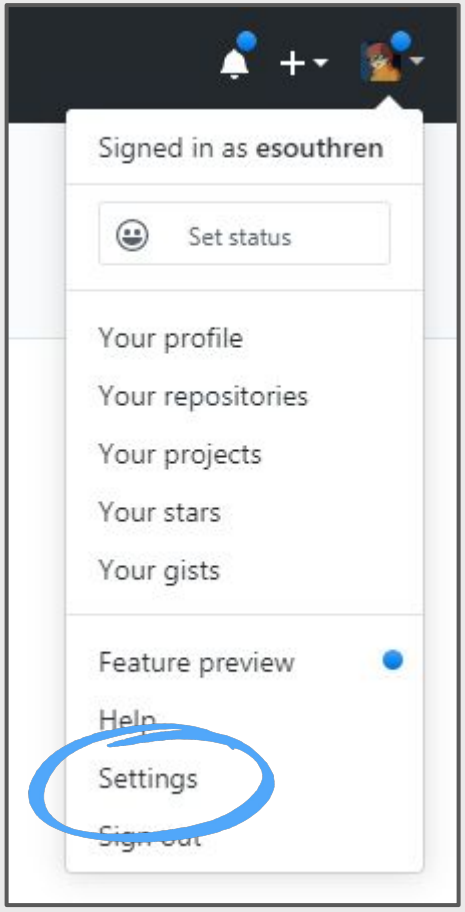
Once you're logged in, head to this URL, where the code is stored:

mlhlocal.host/icon-casino-starter

Then, click **Fork** in the the top right of the page. This will create a copy of the code under your username!



Creating an Access Token

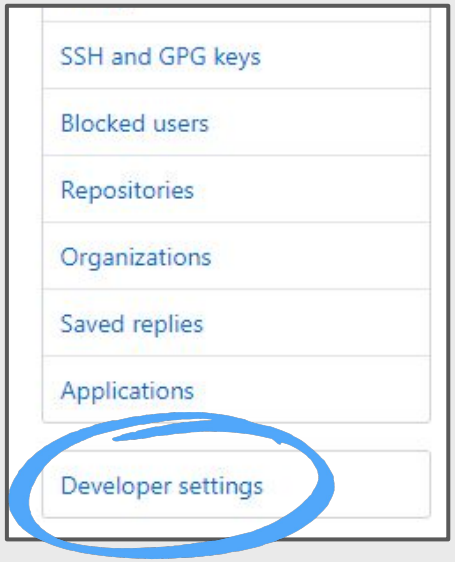


Now that we have a copy of the code, we need to give permission for Morpheus Labs to access it.

We do this by creating an **Access Token**: a secret key which will allow Morpheus Labs to access to our repositories.

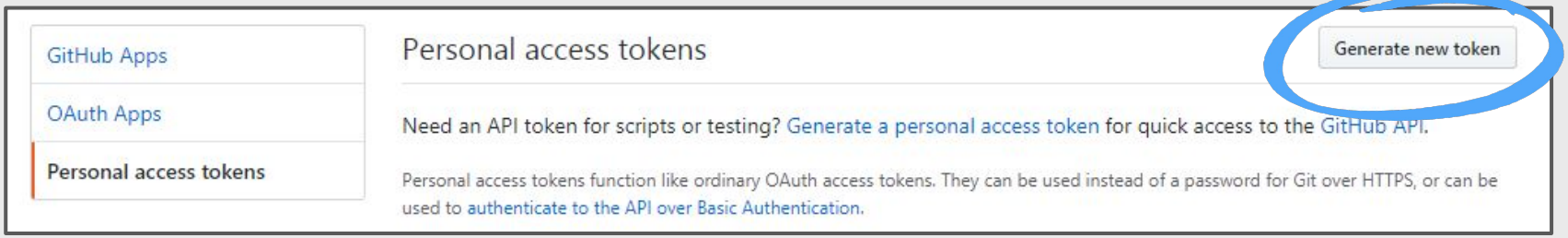
To create one, first click your profile picture and then click **Settings**.

Creating an Access Token



At the bottom of the list of settings, click **Developer Settings**.

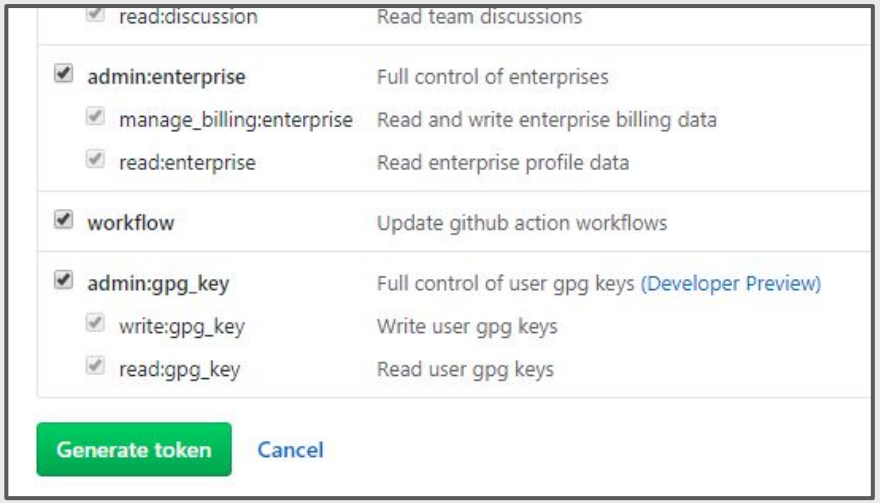
Under Personal access tokens, click **Generate new token**.



Creating an Access Token

Give your new token a note (name) such as **'Morpheus Labs'** to identify it.

Tick all of the boxes and click **Generate token**.



Copy the new token to your clipboard by clicking this icon.

Adding your Access Token

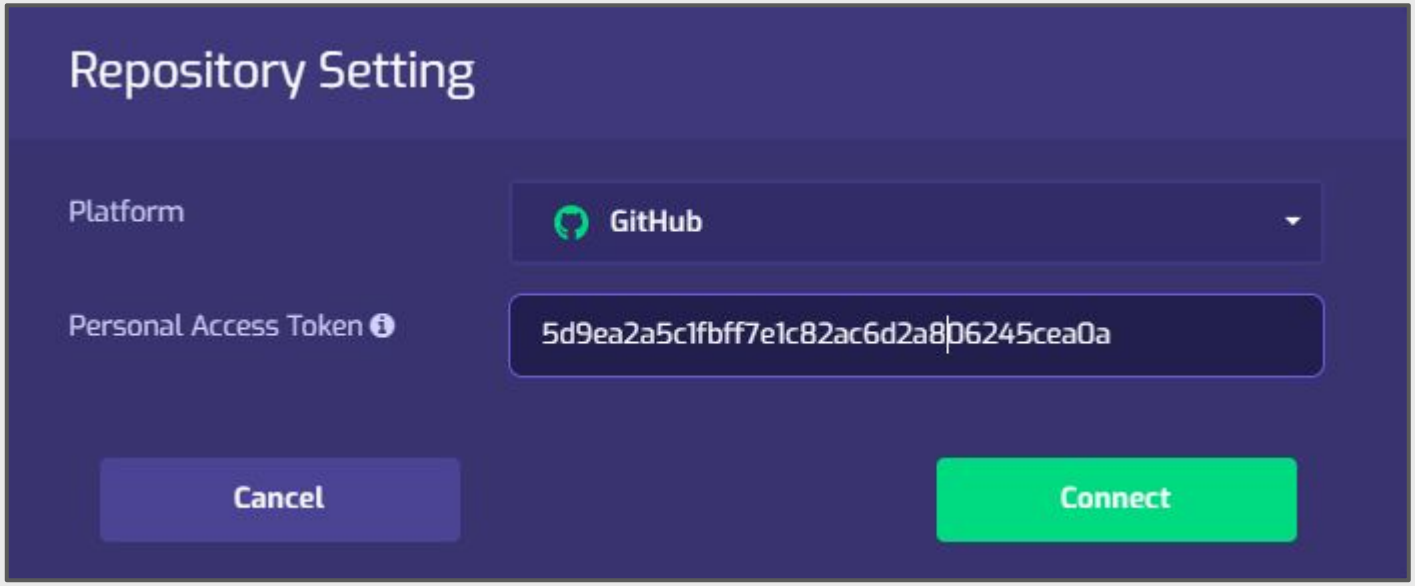
Now we'll add your Access token to Morpheus Labs.

Select the **Dashboard** and click **+ Repository Connection**.

Adding your Access Token

Select **GitHub** as the Platform, and paste in your GitHub access token.

Then click **Connect**, and then **Save**!

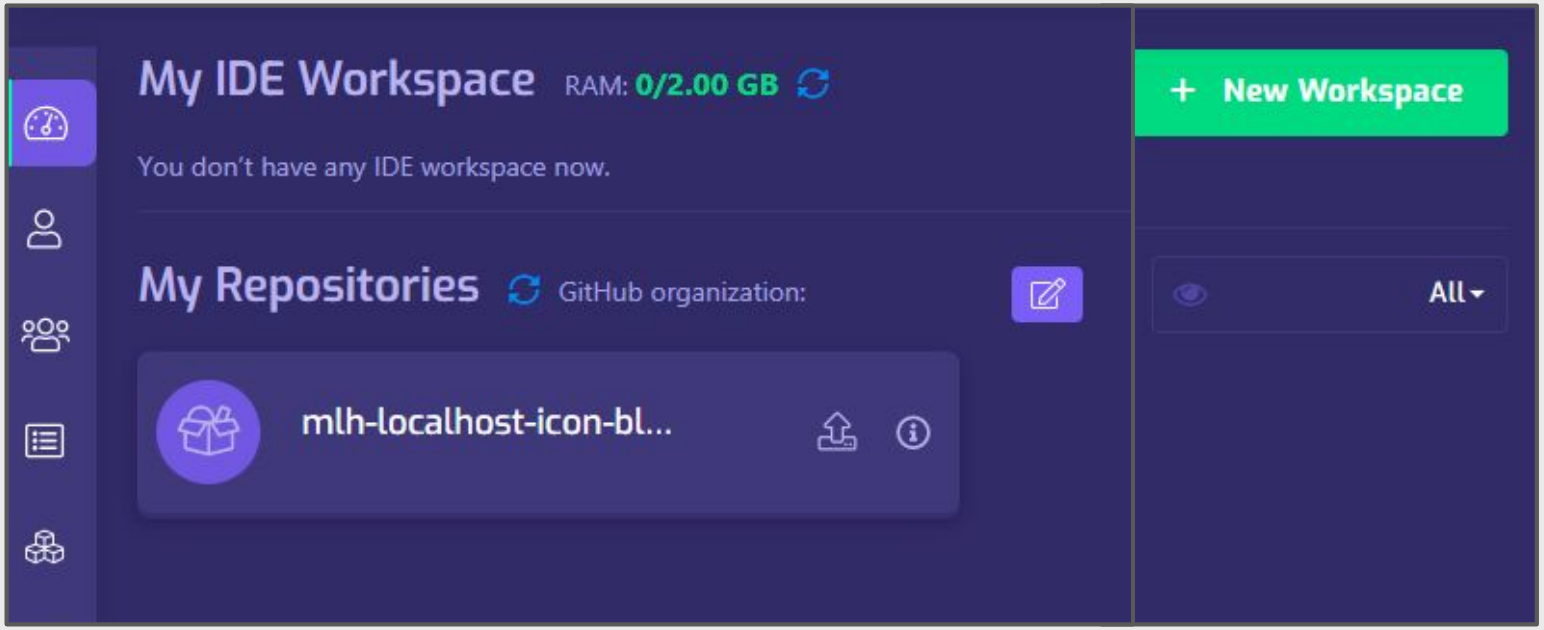


Creating a Workspace

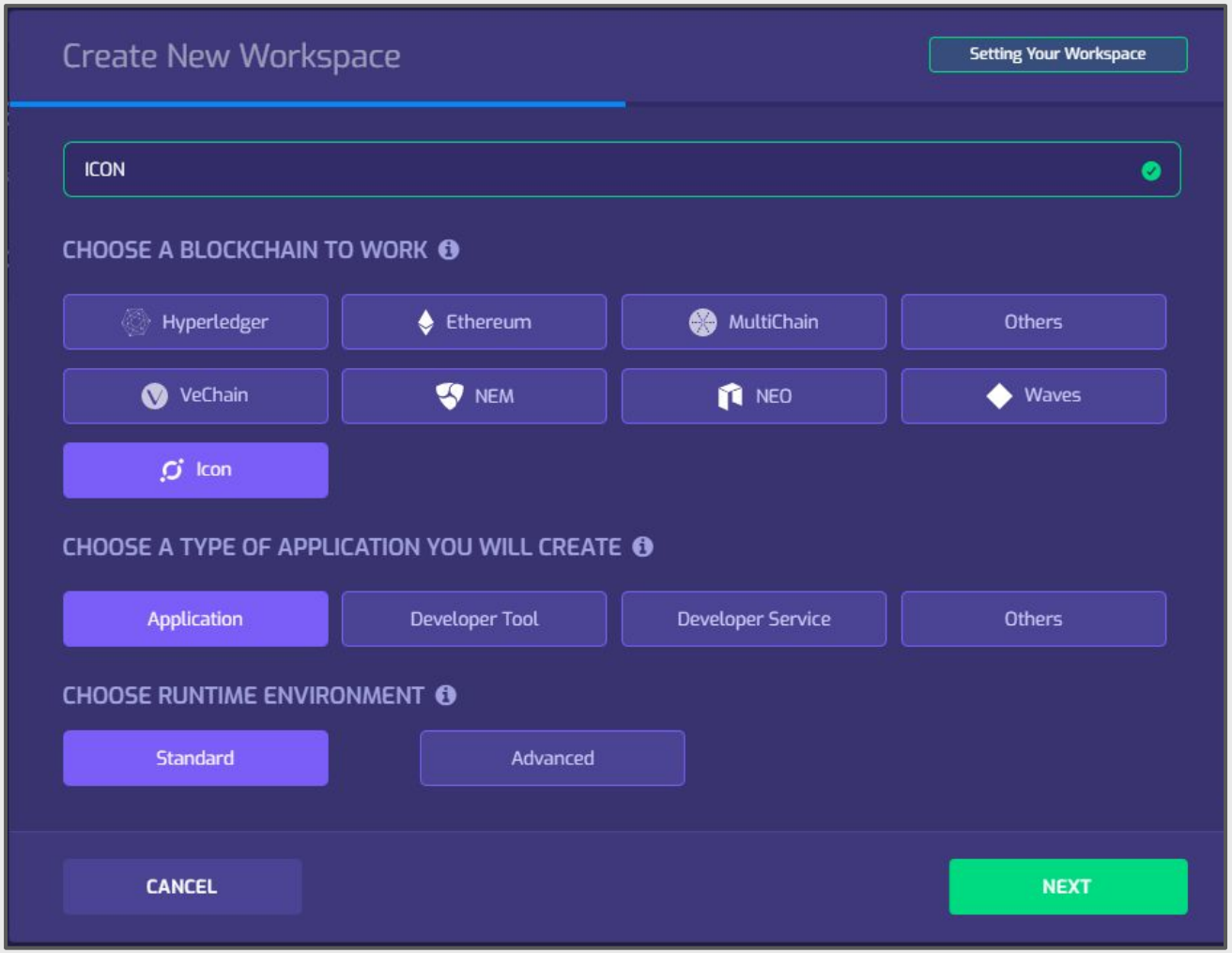
Now you can see your forked repository.

Now, we can create a Workspace and start writing some code!

Click **New Workspace**.



Creating a Workspace

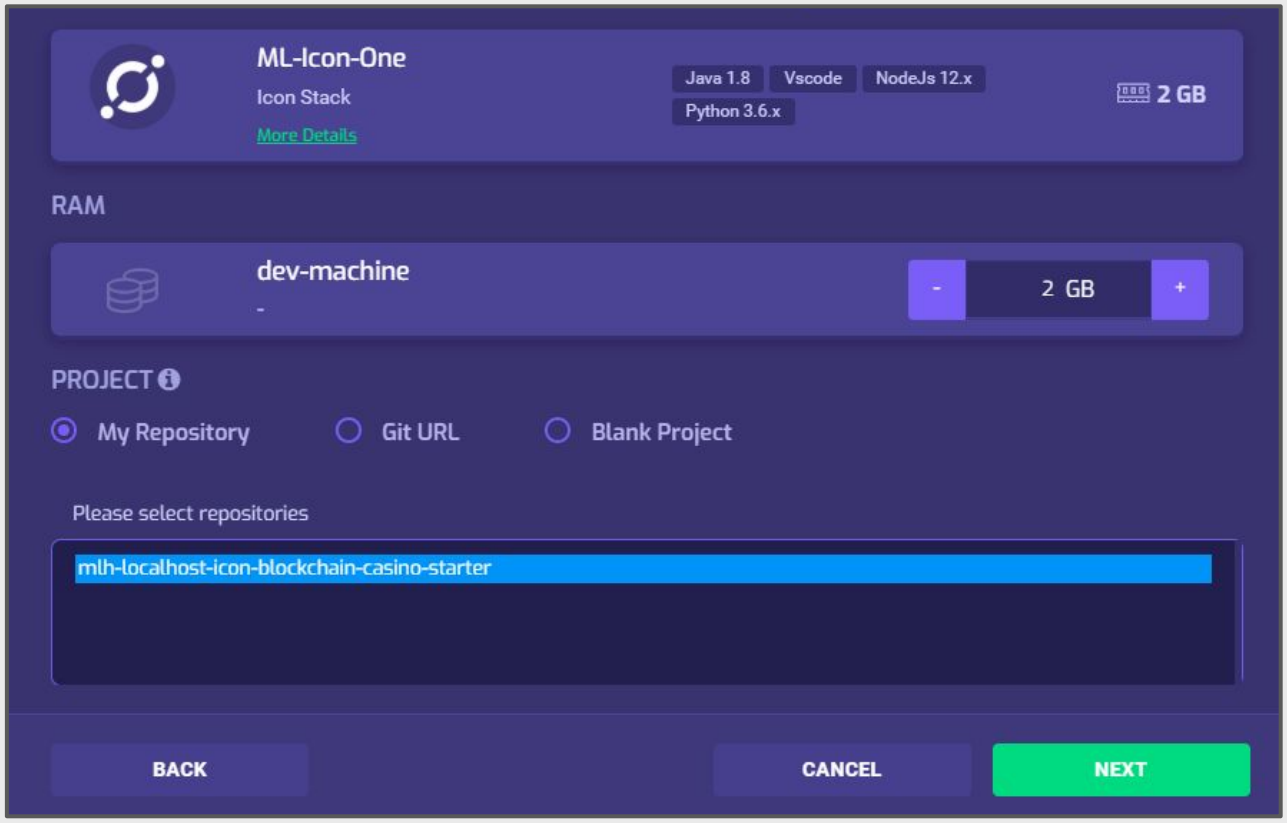


Give your workspace a name, select the ICON Blockchain, then click **Next**.

Creating a Workspace

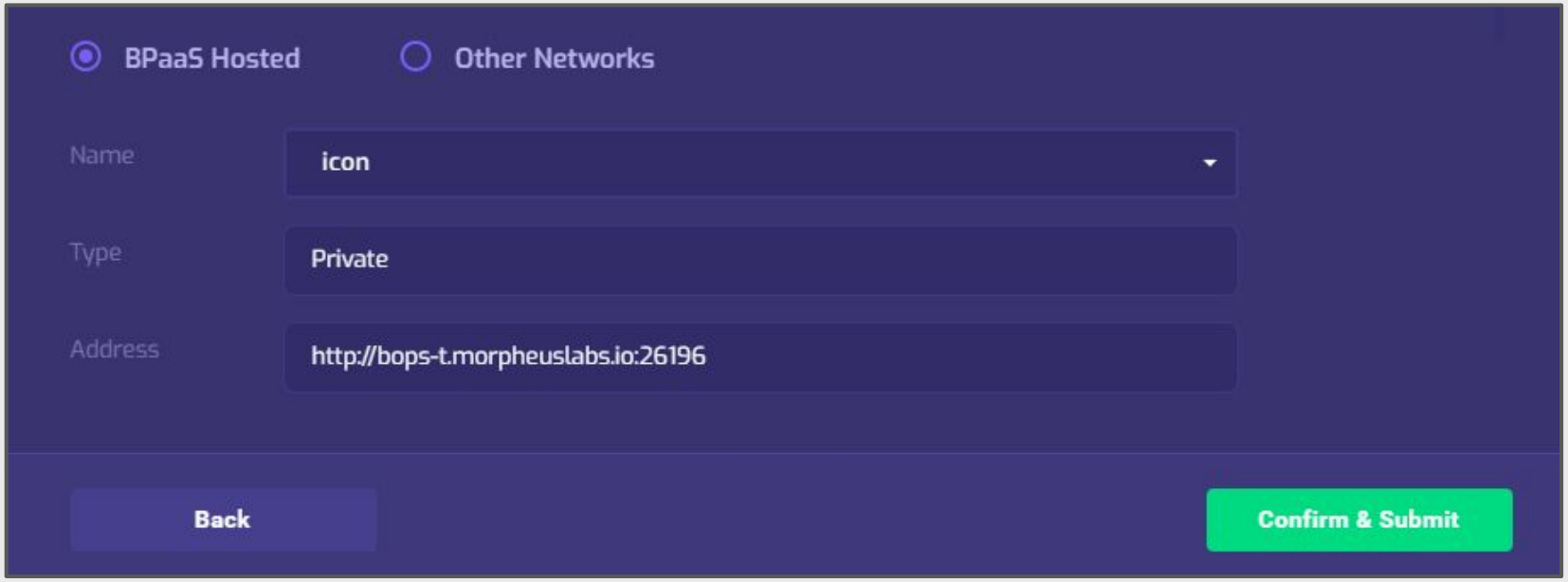
Leave the stack and RAM settings as default.

Select your starter repository then click **Next**.



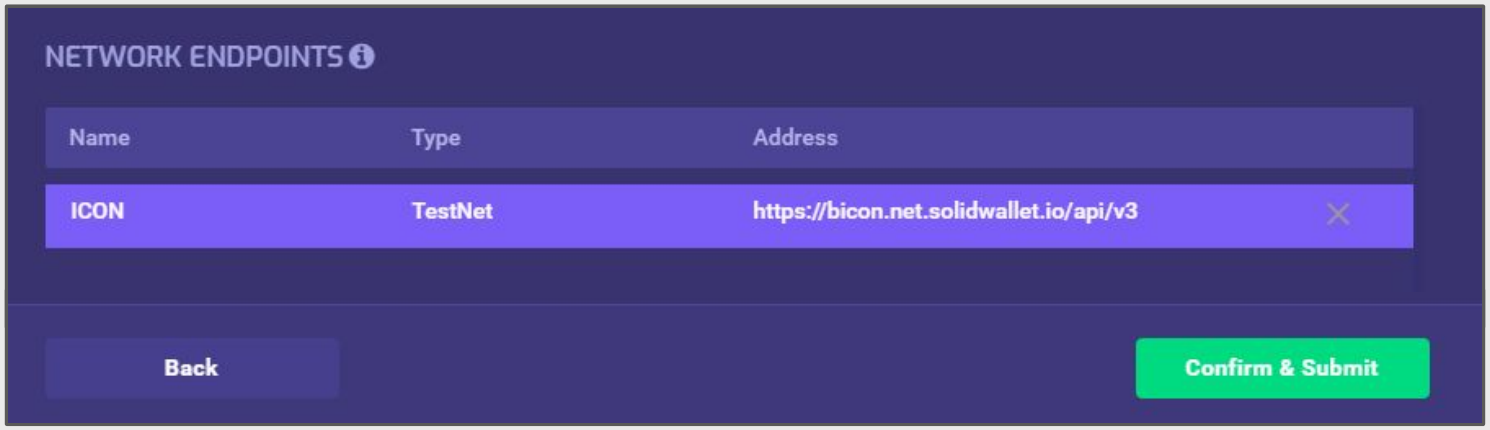
Creating a Workspace

Keep the default **BPaaS Hosted** option selected, ensure that Name is set to **ICON**, and click **Confirm**.



Creating a Workspace

Select your newly created network, then click **Confirm & Submit**.



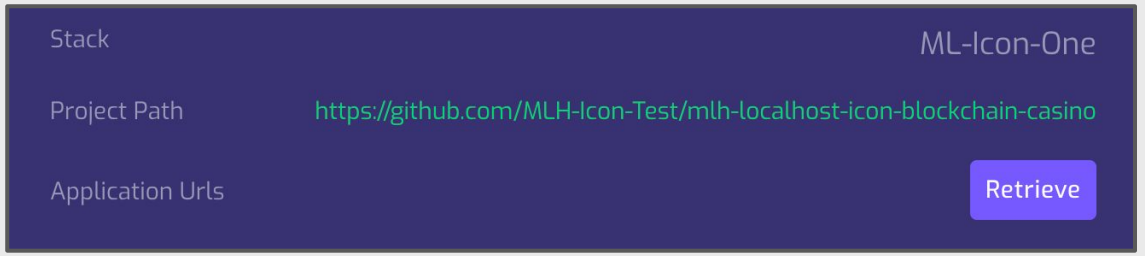
Click the > Icon on your workspace to start it.

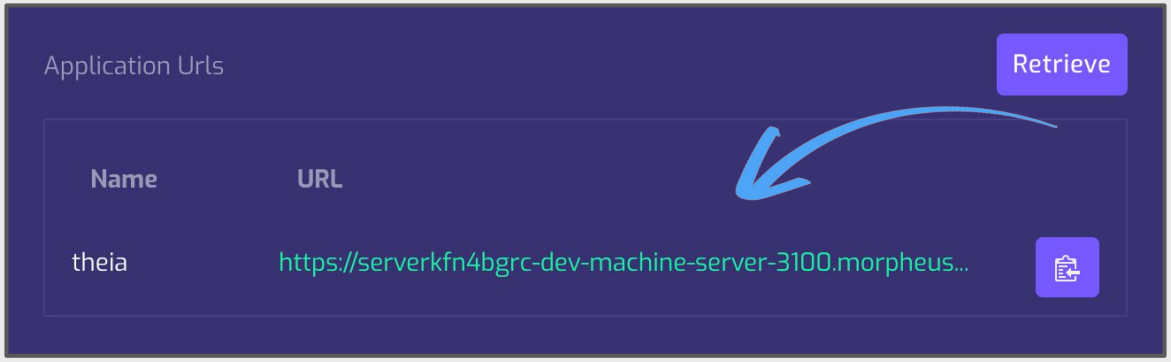
Wait until the icon turns green.

Open the Workspace

Select the **i** icon.

Click **Retrieve**.





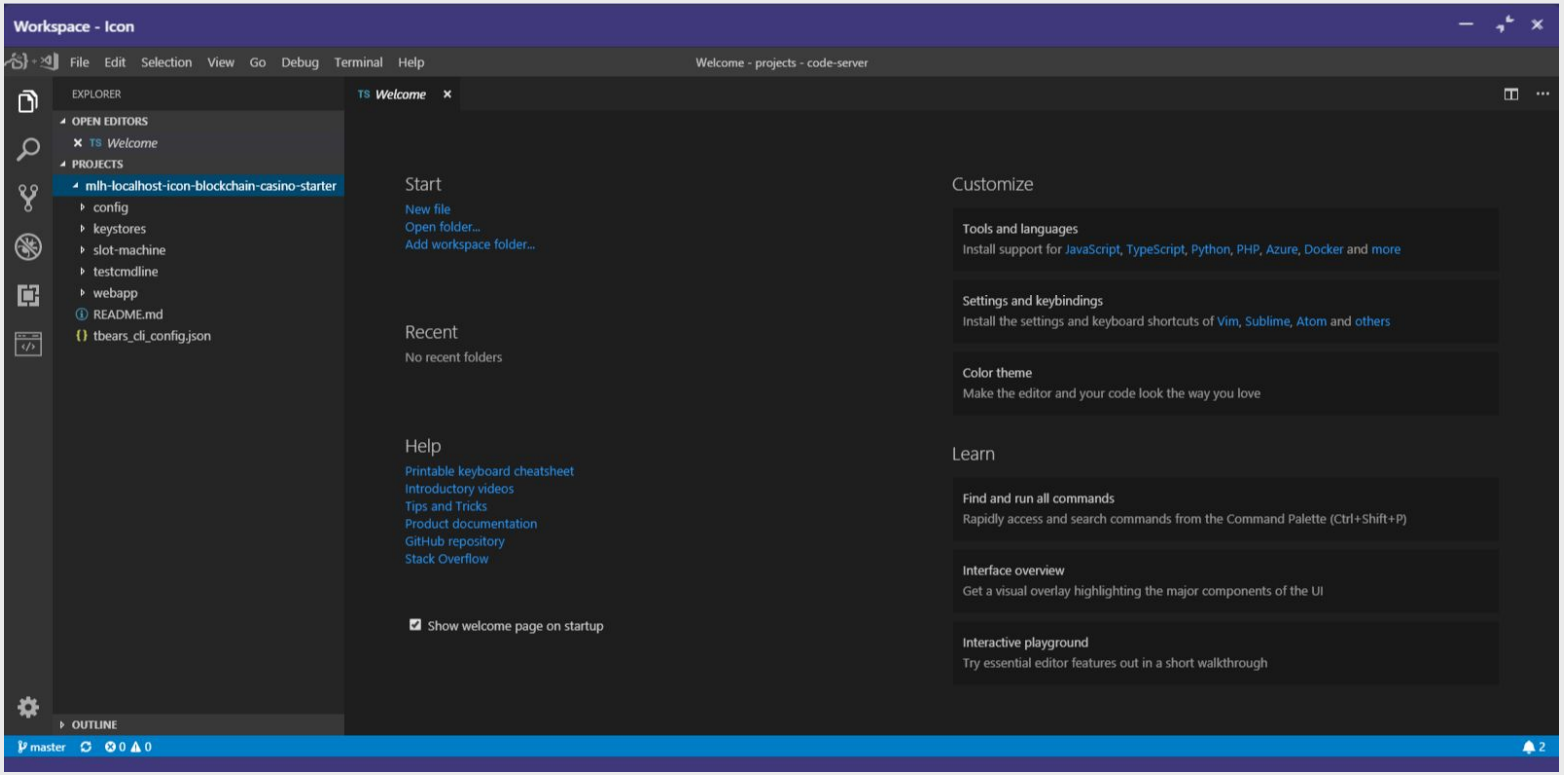
Click the URL!

Let’s recap!

We’ve configured our Morpheus Labs workspace.

Now, we’re ready to start writing some code!

Welcome to your Workspace! This is a **Visual Studio Code** environment.



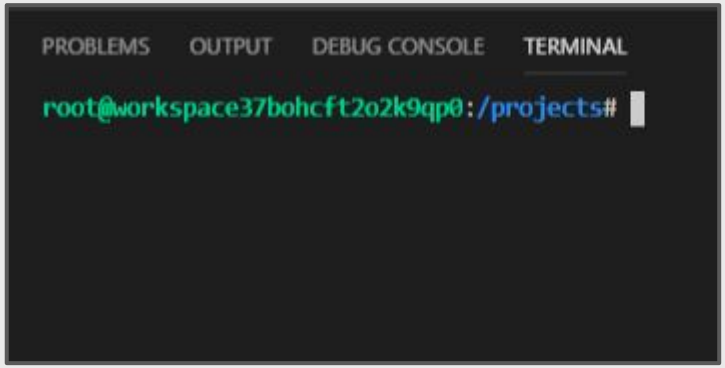
Let’s dive in and write some code!

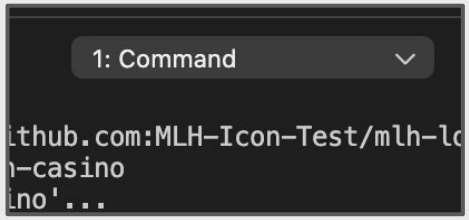
Exploring the Workspace

Let's check out the web application by running it.

Open a Terminal window by clicking **Terminal -> New Terminal**, or typing **Ctrl+Shift+'**.

This is a Terminal window, where we can run commands to launch our web application.



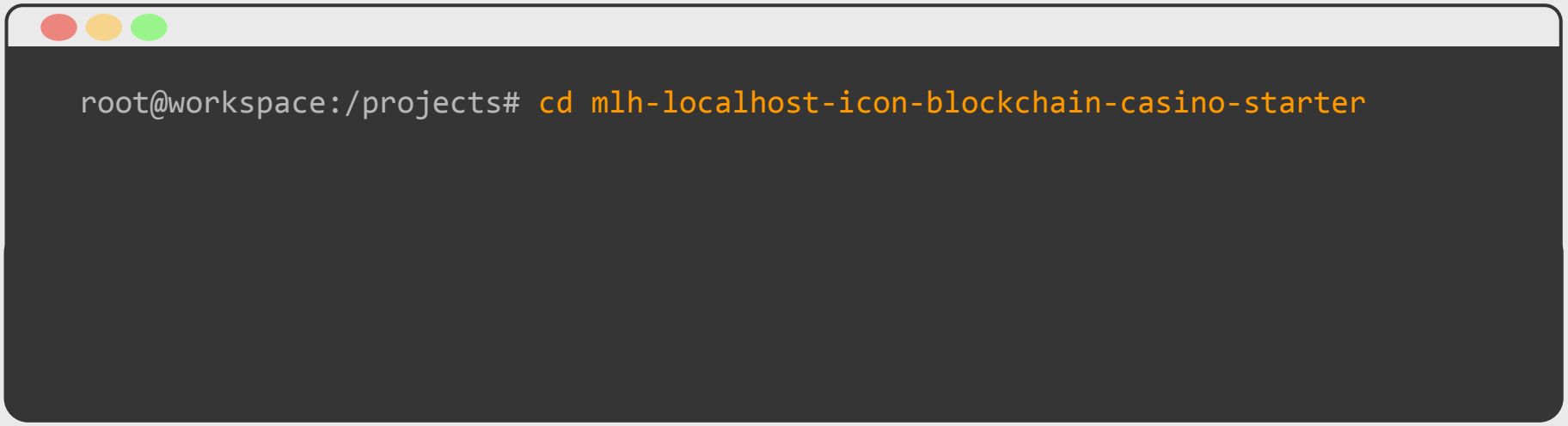


Make sure the dropdown menu in the terminal window is set to **Command**.

Exploring the Workspace

Type the following two commands into the Terminal window, pressing Enter after each command.

We first change directory (**cd**) into the source code folder.



Writing a SCORE

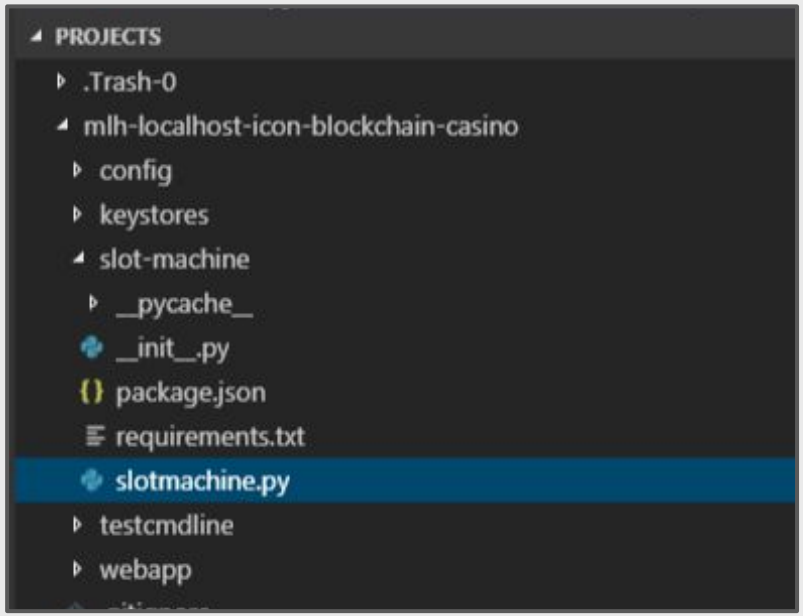
Before we deploy our SCORE to the ICON network, we're going to add some code which will determine whether a player has won the game or not.

SCOREs are coded in Python - we're going to add a new function to our contract called **play**.

Let's get started!

Writing a SCORE

Head back to your workspace tab.



Navigate to the **slot-machine** folder and open the **slotmachine.py** file by clicking on it.

Anatomy of a SCORE

SCOREs are Python programs. They contain functions which can be used externally, like from our web app. Take a look at the code of **slot_machine.py**. What **@external** functions do we have?

set_treasury() **get_treasury()**	These functions add ICX to the SCORE treasury, and retrieve the current treasury balance.
play()	This is used when we 'play' the slot machine. We'll be adding code here!
get_results()	Returns the result of a slot machine spin.

Writing a SCORE

We're going to add code to the **play** function, starting on **line 46**.

There's a couple of things this function is going to do:

- Check that the bet is valid, and that the treasury has enough ICX to pay out if the player wins.
- Play the game by randomly generating numbers.
- If it's a win, send an ICX transaction to the player.
- Send the results back to the webapp to be displayed.

Writing a SCORE

Let's start by getting the values we need.

```
45  def play(self):
46      amount = self.msg.value
47      balance = self.icx.get_balance(self.address)
48      Logger.info(f"Current Balance {balance}.", TAG)
49
```

Line 47: Retrieves the bet value from the transaction that the SCORE receives.
Line 48: Gets the ICX balance of the treasury.
Line 49: Log (display) this information to the console.

Writing a SCORE

Next, let's check that the bet is valid.

```
50    if balance <= amount * PAYOUT_MULTIPLIER:
51          revert(f"Balance {balance} not enough to pay a prize.")
52
53
```

Lines 51-53: Checks that the treasury has enough ICX to pay the player if they win. If not, exit the function and display an error message.

Writing a SCORE

Next, let's check that the bet is valid.

```
53    if amount <= 0 or amount > 10 ** 24:
54          revert(f"Betting amount {amount} out of range.")
55
56    payout = min(amount * PAYOUT_MULTIPLIER, balance)
```

Lines 53-54: Checks that the bet amount is in a valid range (between 0 - 10^{24}). If not, display an error message.

Line 56: Calculates the amount to pay the winner: Either the bet amount multiplied by the payout_multiplier (you can see the value of this variable on **Line 5**), or the entire treasury balance - whichever is smaller.

Writing a SCORE

Here's the magic: deciding if the player has won or not.
In order to make the game **provably fair**, we want to generate a truly random number.

```
58  seed = (str(bytes.hex(self.tx.hash)) + str(self.now()))
59
```

Line 58: There's a lot happening in this line, so let's break it down.
We take the transaction hash **(self.tx.hash**) and convert it from hexadecimal to bytes. We then convert that into text using the **str** function.

self.now() gives us a string of the current time: something that will be different every time the game is played.

We create a string called **seed** by adding these two strings together.

Writing a SCORE

We want to give the player a 1 in 10 chance of winning.
An easy way to achieve this from a random number is to say 'Can we divide this number by 10 evenly?'

```
59   result = (int.from_bytes(sha3_256(seed.encode()), "big") % 10)
60   win = (result == 0)
```

Line 59: We take the **seed** string and encode it into a secure hash using the **sha3_256** hashing function. We convert this into an integer (a number) using **int**. We then make use of the **Modulo (%)** operator. It means 'take this number, divide it by 10, and return the remainder'.

Line 60: If the result equals 0, this means our randomly generated number can be divided by 10 cleanly, and thus ends in 0. If this is the case, win equals **true** and the player has won. For all other results, win is false.

Writing a SCORE

Now we prepare the results to send back to the webapp.

```
62    json_result = {
63          "index": self.tx.index,
64          "nonce": self.tx.nonce,
65          "from": str(self.tx.origin),
66          "timestamp": self.tx.timestamp,
67          "txHash": bytes.hex(self.tx.hash),
68          "amount": amount,
69          "result": win,
70    }
71
72    self._play_results_array.put(str(json_result))
73
```

Lines 62-70: Prepares a message to send back to the user.
Line 72: Places the message into an array to be read by the function **get_results** on **Line 90**.

Writing a SCORE

Let's send the winnings back to the user if they have won.

```
75    if win:
76        Logger.info(f"Amount owed to winner: {payout}", TAG)
77        try:
78            self.icx.transfer(self.msg.sender, payout)
79            self.FundTransfer(self.msg.sender, payout, False)
80            Logger.info(f"Win!. Sent winner ({self.msg.sender}) {payout}.", TAG)
81        except:
82            Logger.info(f"Problem. Winnings not sent. Returning bet.", TAG)
83
84    else:
85        Logger.info(f"Player lost. ICX retained in treasury.", TAG)
86
```

Lines 75-82: If the player has won, attempt (try) to send them an ICX transaction. If it fails (except), display an error message to the user.
Line 86-87: If they haven't won, display a message to the user.

Code Recap

```
45      def play(self):
46              amount = self.msg.value
47              balance = self.icx.get_balance(self.address)
48              Logger.info(f"Current Balance {balance}.", TAG)
49
50              if balance <= amount * PAYOUT_MULTIPLIER:
51                   revert(f"Balance {balance} not enough to pay a prize.")
52
53              payout = min(amount * PAYOUT_MULTIPLIER, balance)
54              seed = (str(bytes.hex(self.tx.hash)) + str(self.now()))
55
56              result = (int.from_bytes(sha3_256(seed.encode()), "big") % 10)
57              win = (result == 0)
58              json_result = {
59                     "index": self.tx.index,
60                   "nonce": self.tx.nonce,
61                   "from": str(self.tx.origin),
62                   "timestamp": self.tx.timestamp,
63                   "txHash": bytes.hex(self.tx.hash),
64                   "amount": amount,
65                   "result": win,
66              }
67
68              self._play_results_array.put(str(json_result))
69
70              if win:
71                  Logger.info(f"Amount owed to winner: {payout}", TAG)
72                  try:
73                      self.icx.transfer(self.msg.sender, payout)
74                      self.FundTransfer(self.msg.sender, payout, False)
75                      Logger.info(f"Win!. Sent winner ({self.msg.sender}) {payout}.", TAG)
76                  except:
77                      Logger.info(f"Problem. Winnings not sent. Returning bet.", TAG)
78
79              else:
80                  Logger.info(f"Player lost. ICX retained in treasury.", TAG)
81
82
83
```

Let’s recap!

We created a workspace finished writing the SCORE.

Next, we need to create an ICON wallet to store the coins we’ll use to play the game!

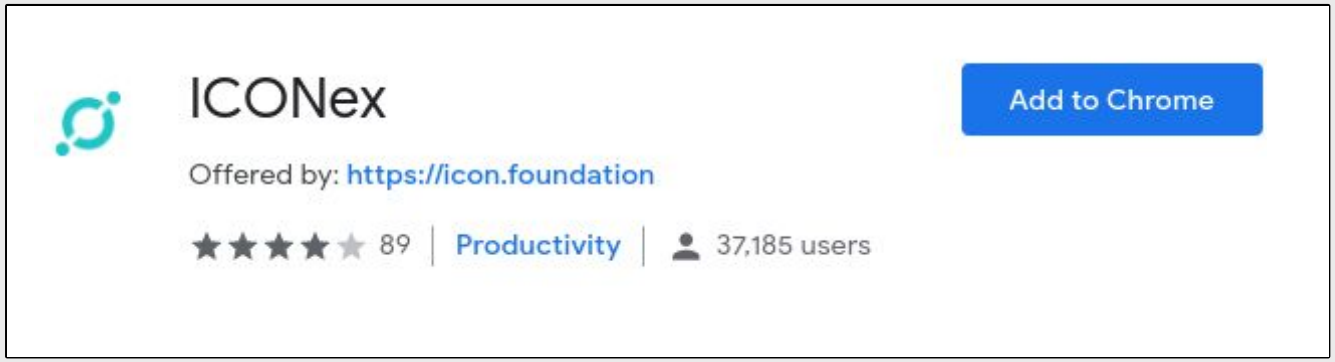
Create an ICON Wallet

Head to this URL and click on **Wallet**.

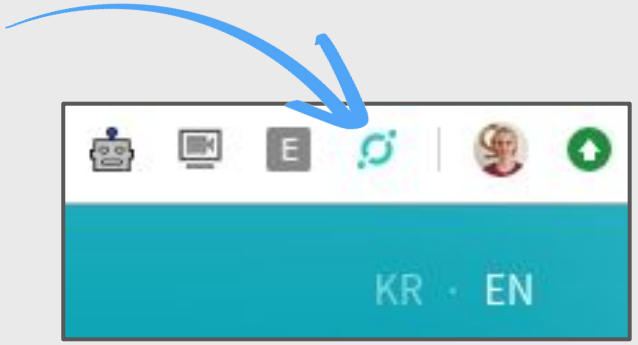
mlhlocal.host/icon

Create an ICON Wallet

Click **Add to Chrome.**

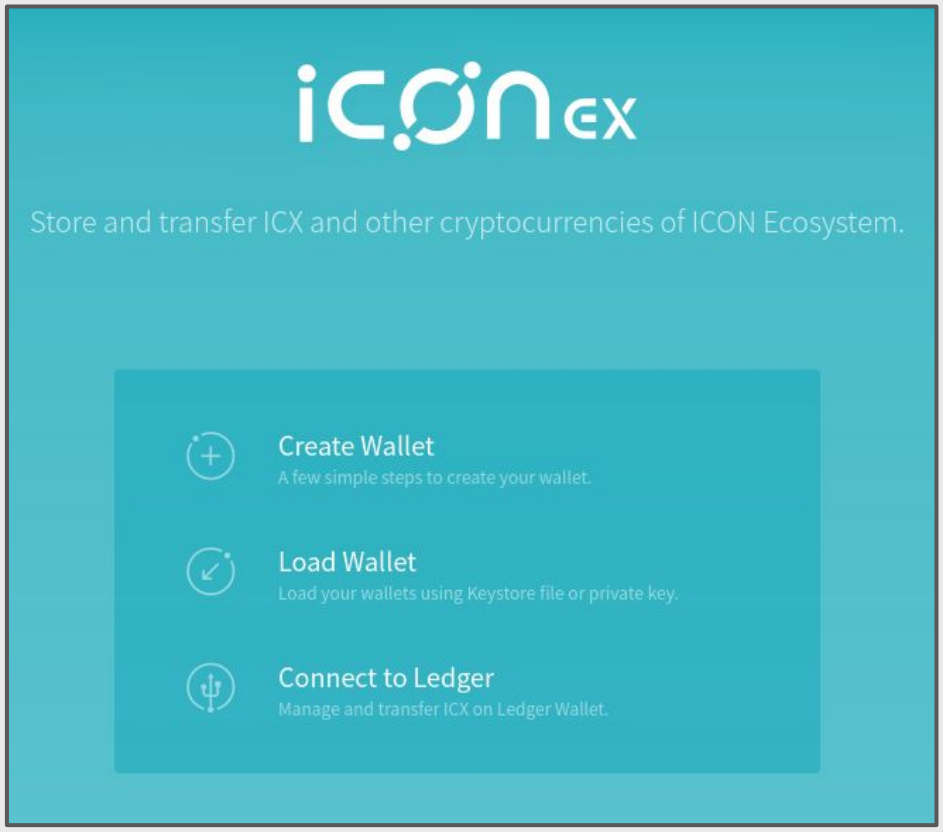


When it has finished installing, click the ICON logo from the Chrome extensions bar.



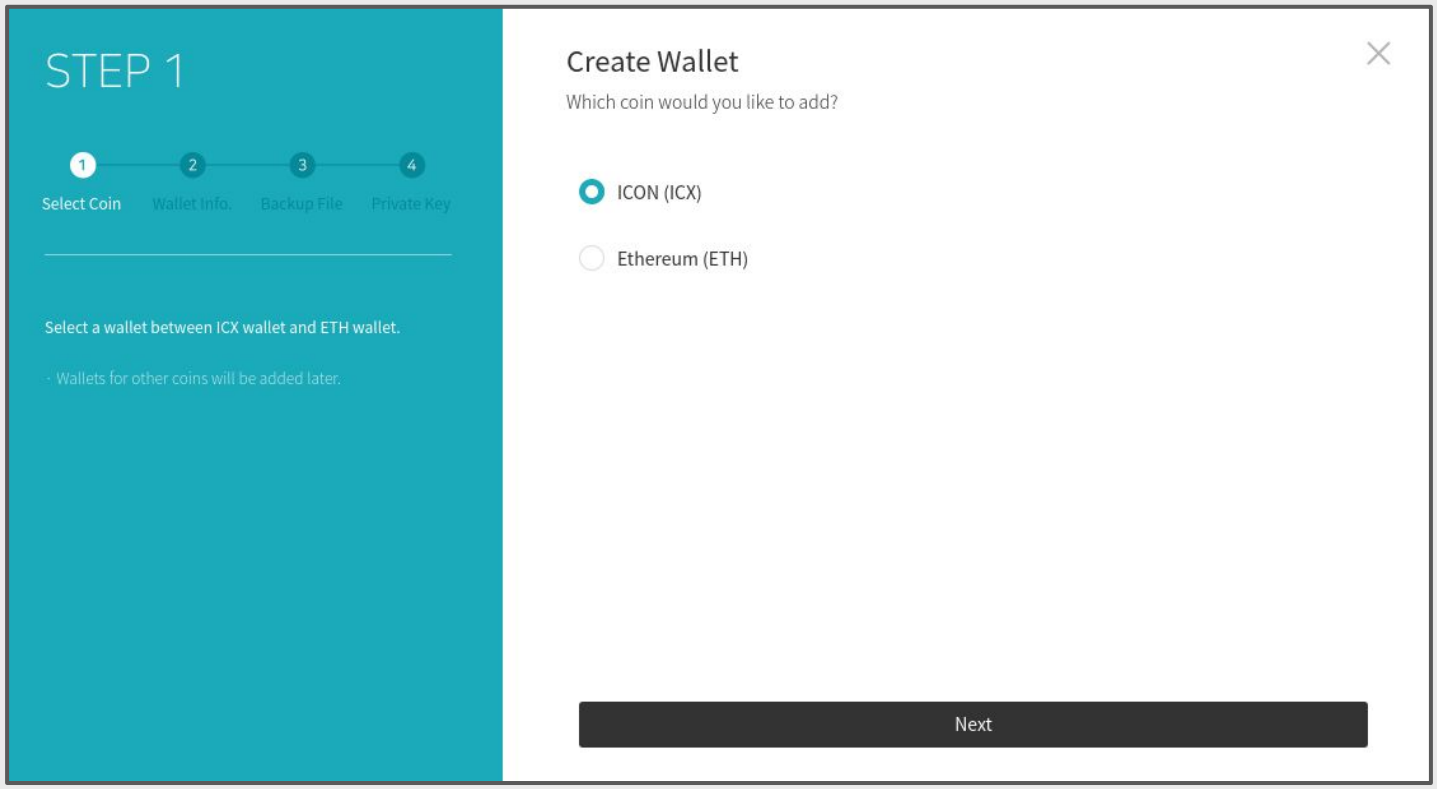
Create an ICON Wallet

Select **Create Wallet**.



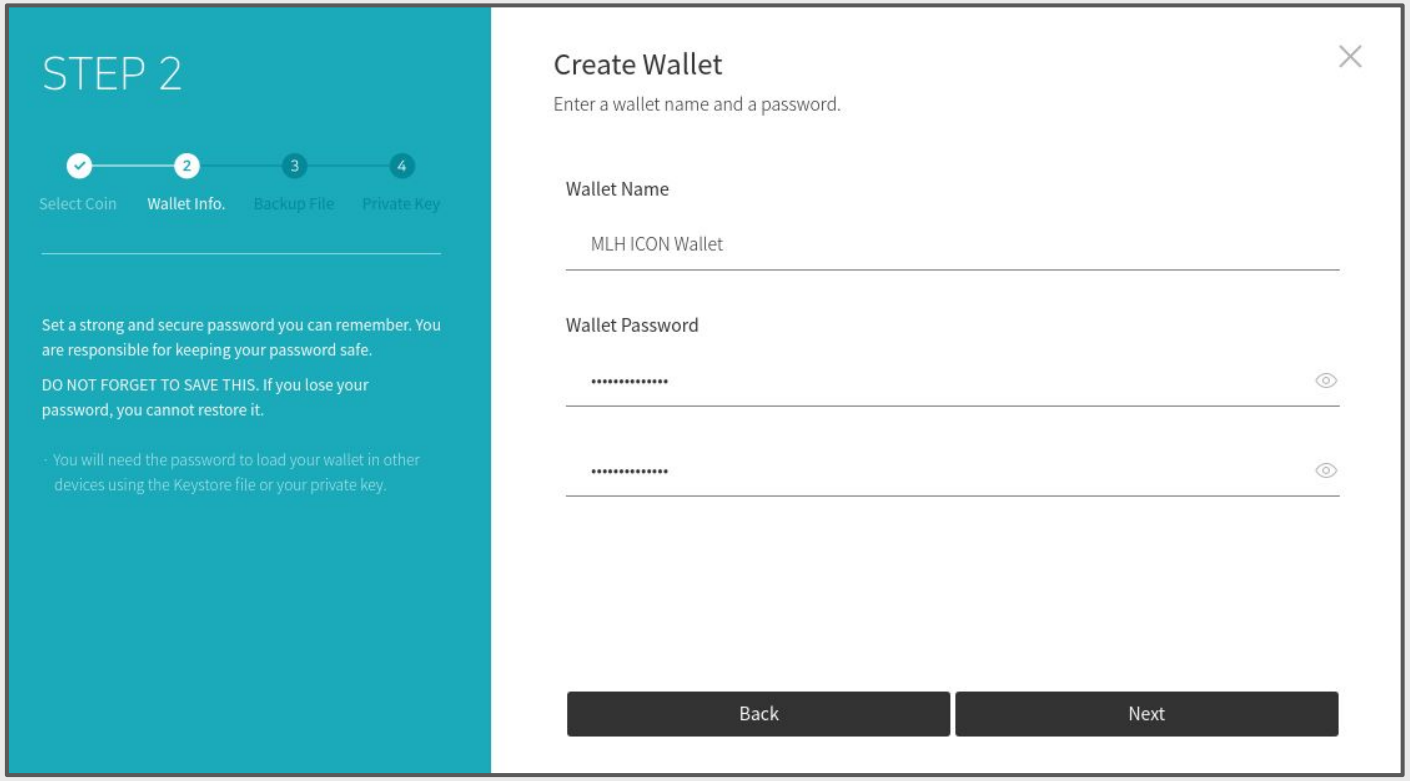
Create an ICON Wallet

Keep the wallet as ICON (ICX) and click **Next**.



Create an ICON Wallet

Give your wallet a name, and secure it with a password. Click **Next**.



Create an ICON Wallet

Click **Download Keystore File**, and then click **Next**.

Key Term

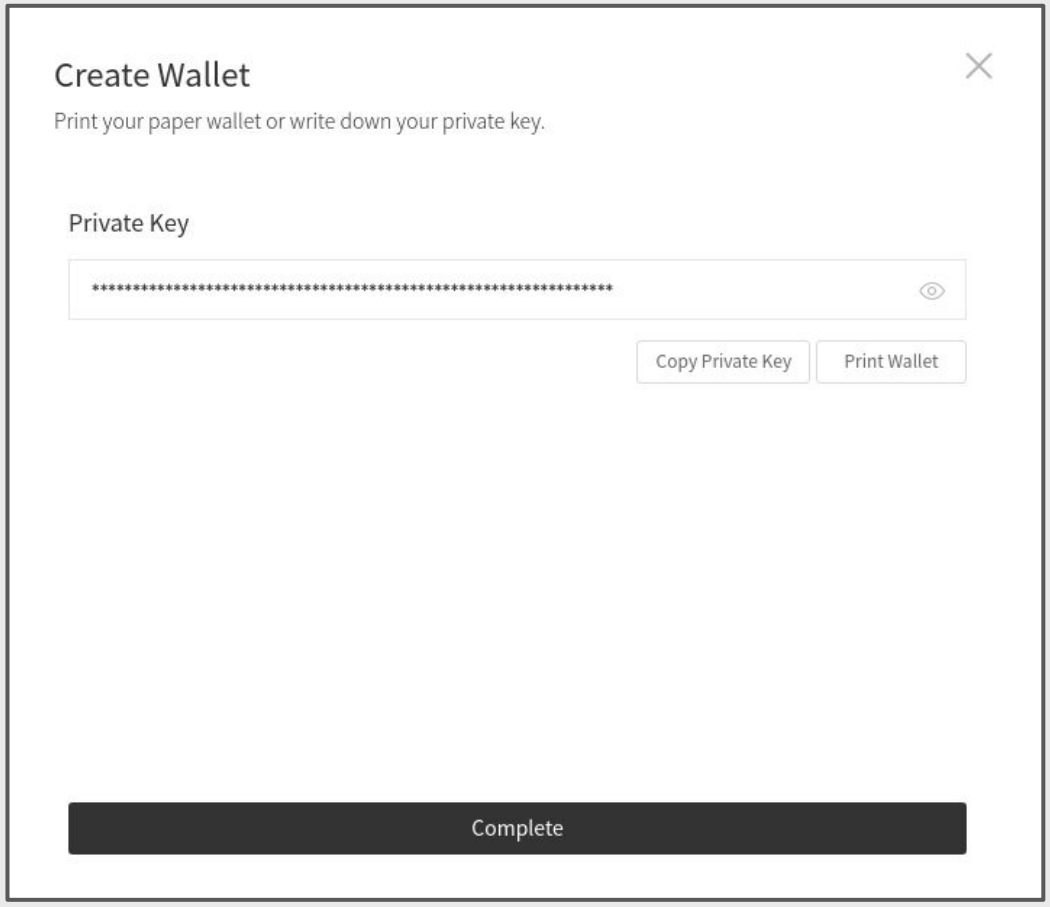
Keystore File: A file which contains an encrypted private key. You can use it to access your wallet from other devices. Be careful not to lose it!

Create an ICON Wallet

Click **Copy Private Key** and make a note of it.

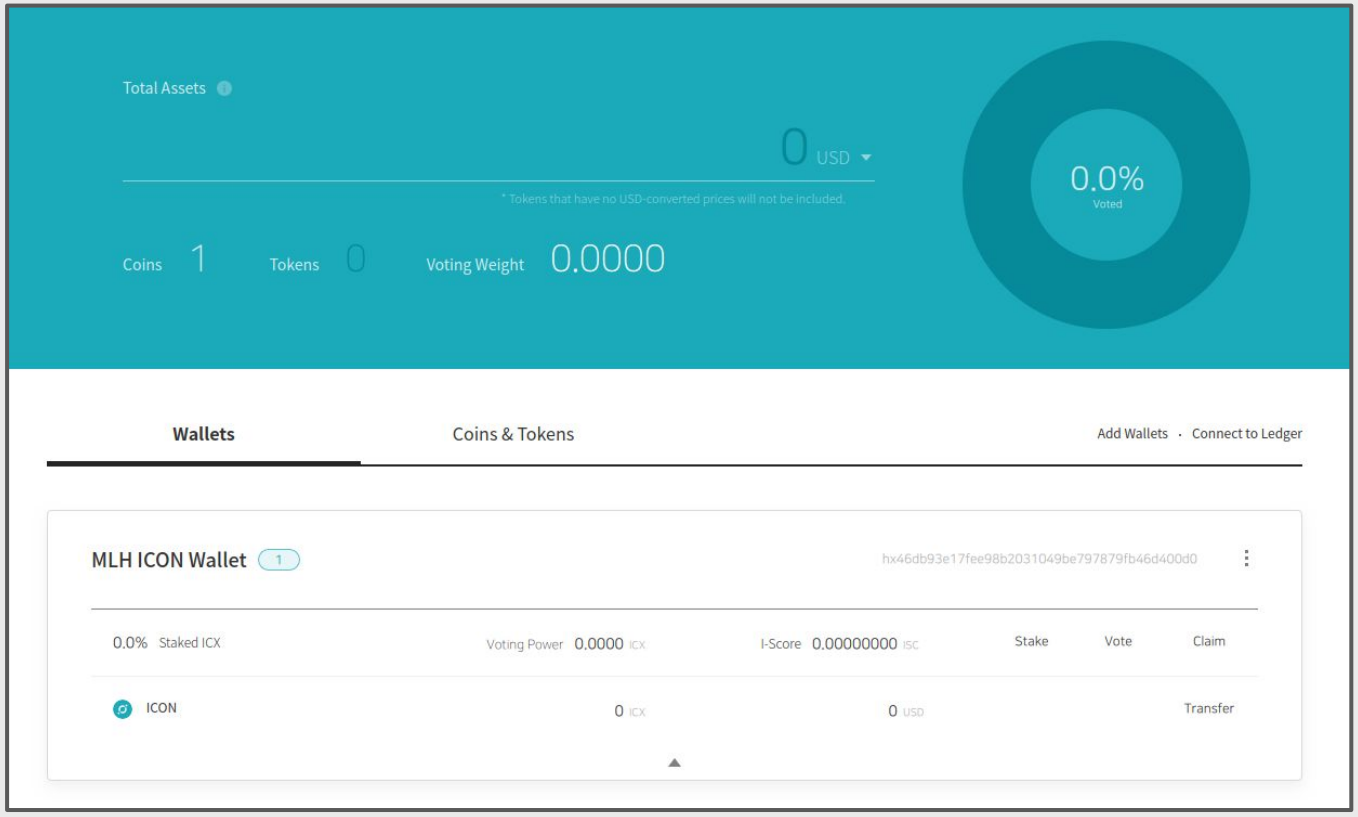
This Key will allow you to access your wallet from devices other than your current one.

Lastly, click **Complete**.



Your Wallet Dashboard

This is your wallet dashboard, where you can view your current number of Coins and Tokens.

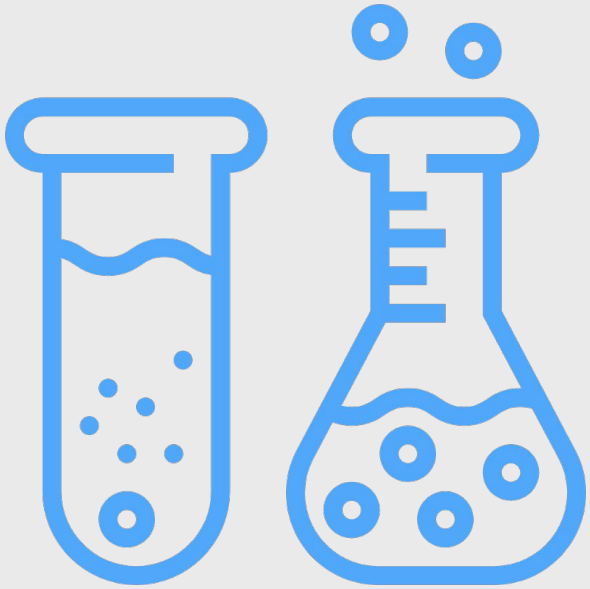


Running your Wallet on TestNet

Currently our wallets are connected to **MainNet** - the main, real life ICON network.

We're going to be running our application on **TestNet**, so we don't spend any real money! We'll make a small change so that your wallet is on **TestNet** instead of **MainNet**.

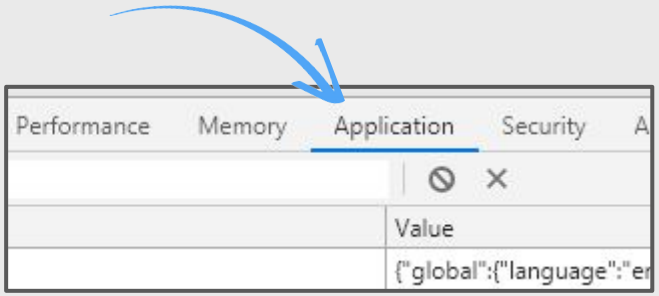
TestNet, as the name suggests, allows us to test SCOREs and Transactions without worrying that we might lose real ICX.

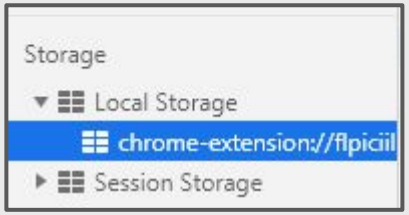


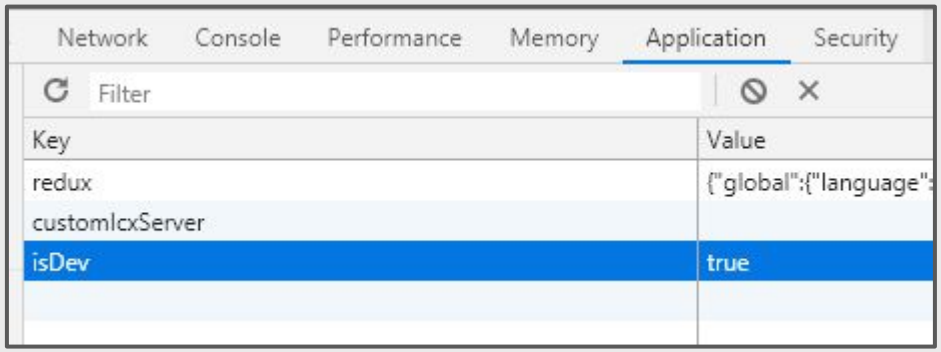
Running your Wallet on TestNet

In Chrome, open your Wallet tab. Press **F12** (or Ctrl+Shift+i) to open the Developer Tools.

Select the **Application** tab. Under Local Storage, select **chrome-extension**.







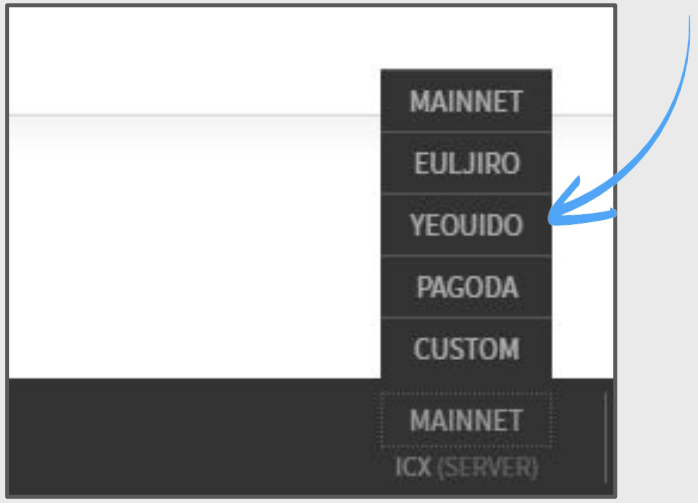
Double click an empty line to add a new key/value pair. Type **isDev** as the key and **true** as the value.

Running your Wallet on TestNet

Reload your tab by pressing **F5**.

At the bottom of your wallet screen will be a new menu which currently says **MAINNET**. Select it and click **YEOUIDO**.

Fun fact: Yeouido is a small island in Seoul, Korea. It's the city's financial district. It is also the codename for ICON's TestNet!



Receive some Test ICX Coins

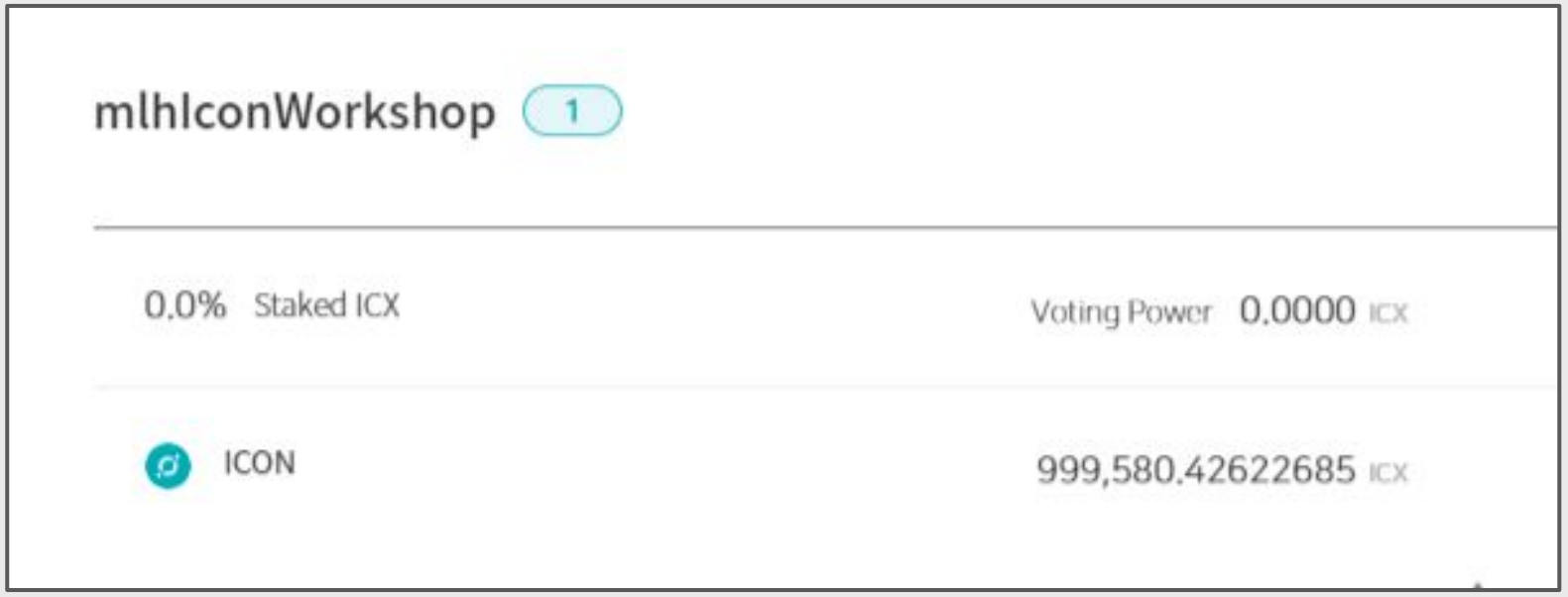
Now that you have a wallet, we're going to give you some test ICX coins to test your app.

Head to this URL and paste your wallet address in the Faucet to get 20 test ICX!

http://mlhlocal.host/faucet

Check your Balance

Refresh your wallet page and you should now have a balance of test ICX coins!

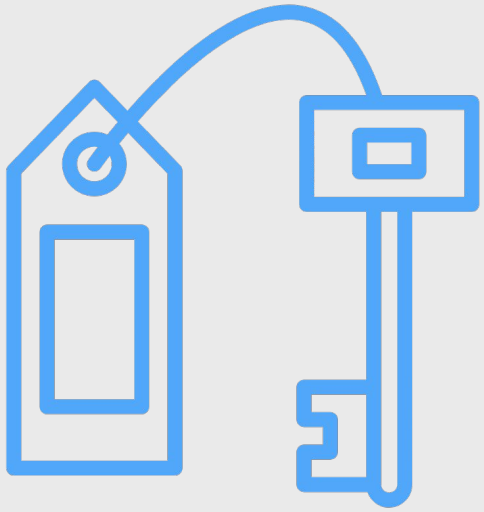


Add your Keystore file

Your downloaded **Keystore File** contains authorization credentials to access your wallet.

This, combined with your wallet password, keeps your ICX protected!

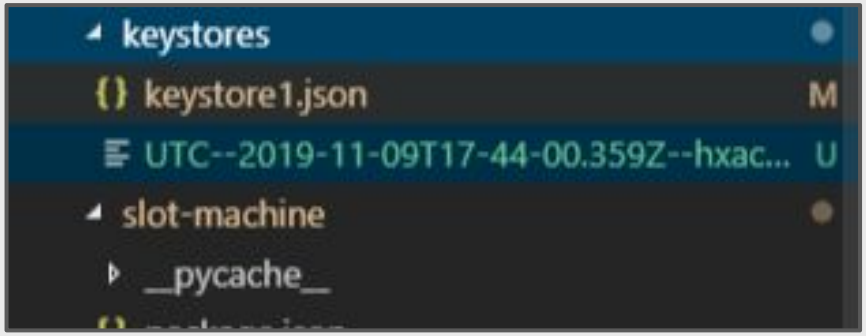
It's important to **never** give your keystore file (or password) to anyone who you don't want to access your wallet.

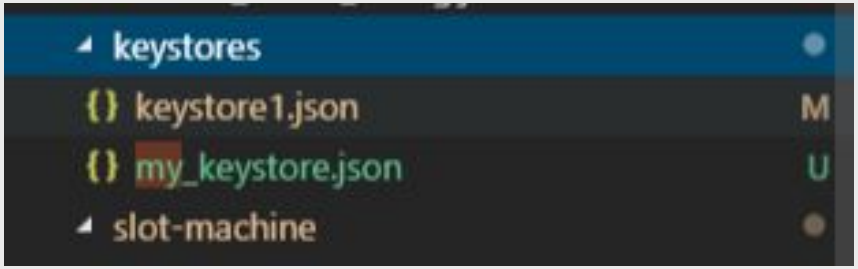


Add your Keystore file

Navigate to the **Keystores** folder in your workspace.

Copy or drag and drop your downloaded Keystore file into this folder. The file will start with **UTC--**.





Rename the file by right clicking on it, and give it the file extension **.json** so that our webapp can read it.

Change Environment Variables

Our source code has an **.env** file, which stores **environment variables** - values that are used across the application.

These can include things like usernames and passwords!

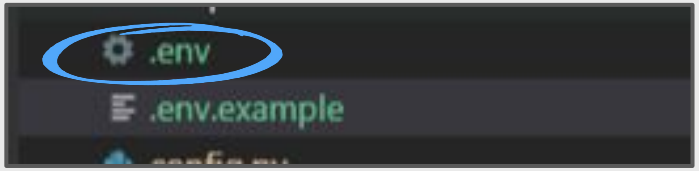
Our source code has a file called **.env.example** in the **webapp** folder. It contains some default values that we will replace, like our wallet password.

Change Environment Variables

Firstly we need to create our own .env file from the .env.example file. If your app is still running, stop it by typing **Ctrl+C** (or Cmd+C on a Mac).

Use the **cp** (copy) command to create a copy of .env.example called .env.

```
root@workspace:/projects/mlh-...-starter# cp webapp/.env.example webapp/.env
```



When the copy is complete, a new file will be visible in the file explorer. **Click on it** and let's modify some values.

Change Environment Variables

We're going to change two values here - the location of your keystore file, and your wallet password.

```
ICON_SERVICE_PROVIDER_URL=https://bicon.net.solidwallet.io/api/v3
CASINO_SCORE_ADDRESS=CX8908459837623409823498734601098 34
PLAYER_WALLET_PRIVATE_KEY_FILE_PATH=./keystores/my_keystore.json
PLAYER_WALLET_PASSWORD=my_secure_password
BET_AMOUNT=10000000000000000000
DEBUG=True
HOST=0.0.0.0
PORT=5000
```

Change the values of **PLAYER_WALLET_PRIVATE_KEY_FILE_PATH** and **PLAYER_WALLET_PASSWORD** to your values, then save the file.

Modifying the tbears Config

One more file to modify: open up **tbears_cli_config.json**.

```
{
  "uri": "https://bicon.net.solidwallet.io/api/v3",
  "nid": "0x3",
  "keyStore": "keystores/<YOUR_WALLET_KEYSTORE_FILE>",
  "from": "<YOUR_WALLET_ADDRESS>",
  "to": "cx0000000000000000000000000000000000000000",
  "stepLimit": "0x50000000",
  ...
```

Change the values of **<YOUR_WALLET_KEYSTORE_FILE>** and **<YOUR_WALLET_ADDRESS>.** Remember, your wallet address starts with **hx**.

Save the file.

Deploying the SCORE

It's time to deploy the SCORE. We can do this straight from the Terminal window using a **tbears** command.

```
root@workspace:/projects/starter# tbears deploy slot-machine

Input your keystore password:

Send deploy request successfully.
If you want to check SCORE deployed successfully, execute txresult command
transaction hash:
0x760029ae75d0b8e3239fefaf9faf12456348b0a522d35e4047806a9730cf5322
```

Let's see if it was successful! If you got an error, try restarting the workspace. **Highlight** the transaction hash - that's the long number that starts with 0x - and copy it by typing **Ctrl+C**.

Checking the SCORE

We'll use another tbears command to verify that the SCORE deployed. Type **tbears txresult** and add **Ctrl+V** to paste in the transaction hash.

```
root@workspace:/projects/starter# tbears txresult 0x760029ae75d0b8e3239fefaf9faf12

Transaction result: {
    "jsonrpc": "2.0",
    "result": {
        "txHash":
"0x760029ae75d0b8e3239fefaf9faf12456348b0a522d35e4047806a9730cf5322",
        "blockHeight": "0x57f70c",
        "blockHash":
"0x7e4f502fff06f869388b41e0036a18f069de5de555200d92c19ce59f122c6b04",
        "txIndex": "0x0",
        "to": "cx0000000000000000000000000000000000000000",
        "scoreAddress": "cx7b5e1a2f68ddfdbca9e2aa16fb66ed2d0b08939f",
        "stepUsed": "0x3e9ace80",
       ...
```

Checking the SCORE

If your output includes a scoreAddress, the deployment was a success!

The address is a unique identifier for our SCORE which now exists on ICON's TestNet.

```
...
    "txIndex": "0x0",
    "to": "cx0000000000000000000000000000000000000000",
    "scoreAddress": "cx7b5e1a2f68ddfdbca9e2aa16fb66ed2d0b08939f",
    "stepUsed": "0x3e9ace80",
...
```

Highlight and copy (**Ctrl+C**) the scoreAddress value.

Let’s recap!

We’ve deployed our SCORE to TestNet!

Next, we need to add some funds to the SCORE so we can play the game.

Modify the .env file

Head back to your **.env** file. Paste the value you got from your T-Bears container and replace the **CASINO_SCORE_ADDRESS** value.

```
ICON_SERVICE_PROVIDER_URL=https://bicon.net.solidwallet.io/api/v3
CASINO_SCORE_ADDRESS=cx7b5e1a2f68ddfdbca9e2aa16fb66ed2d0b08939f
PLAYER_WALLET_PRIVATE_KEY_FILE_PATH=./keystores/my_keystore.json
PLAYER_WALLET_PASSWORD=Mlh_icon_Bl0ckchain
BET_AMOUNT=10000000000000000000
DEBUG=True
HOST=0.0.0.0
PORT=5000
```

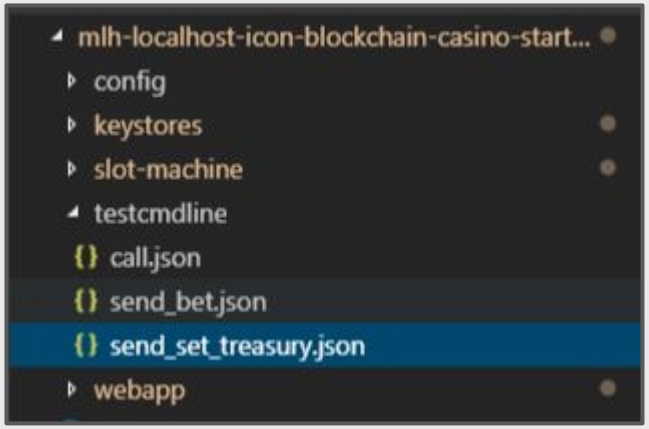
Send in your SCORE Address

After modifying the .env file, out this quick form and copy and paste your score address here:

mlhlocal.host/SCOREaddress

So our friends at ICON can see your smart contract!

Add funds to the Treasury



We're going to send some funds to the SCORE to act as the 'treasury' for the game. We do this by sending a Transaction from our wallet to the SCORE.

We need to modify some details in the transaction, so open up **send_set_treasury.json**.

Add funds to the Treasury

There are two values to modify here: **YOUR_WALLET_ADDRESS** and the **SCORE_ADDRESS**. Grab them from your **.env** file and paste them in here.

Remember, your wallet address starts with **hx** and the SCORE address with **cx**.

Save the file!

```
{
  "jsonrpc": "2.0",
  "method": "icx_sendTransaction",
  "params": {
    "version": "0x3",
    "from": "<YOUR_WALLET_ADDRESS>",
    "value": "0xfffff05b59d3b2",
    "stepLimit": "0x200000",
    "nid": "0x3",
    "nonce": "0x2",
    "to": "<SCORE_ADDRESS>",
    "dataType": "call",
    "data": {
      "method": "set_treasury"
    }
```

Execute the Transaction

We can send this transaction with the tbears command **sendtx**.

In a Terminal window, run the following command:

```
root@workspace:/projects/starter# tbears sendtx send_set_treasury.json

Input your keystore password:

Send deploy request successfully.
If you want to check SCORE deployed successfully, execute txresult command
transaction hash:
0x760029ae75d0b8e3239fefaf9faf12456348b0a522d35e4047806a9730cf5322
```

Test Transaction Success

Let's see if your transaction was successful - if it was, your SCORE will have some ICX coins in it. We can use another tbears command, **balance**, to check the balance of the contract.

Enter the following command with your own SCORE address:

```
root@workspace:/projects/starter# tbears balance cx7b5e1a2f68ddfdbca9e2aa16fb66ed

balance in hex: 0xfffff05b59d3b2
balance in decimal: 72057526851064754

root@workspace:/projects/starter#
```

Table of Contents

Launching the WebApp

Type the following two commands into the Terminal window, pressing Enter after each command.

We first change directory (cd) into the source code folder.

```
root@workspace:/projects/mlh-localhost-icon-blockchain-casino-starter# pip3
install -r webapp/requirements.txt

Successfully installed Flask-1.1.1 Jinja2-2.10.3 MarkupSafe-1.1.1
Werkzeug-0.16.0 iconsdk-1.1.0 itsdangerous-1.1.0 python-dotenv-0.10.3
```

Launching the WebApp

Finally, we launch the app using the **python3** command.

```
root@workspace:/projects/mlh-localhost-icon-blockchain-casino-starter# python3
webapp/main.py

* Serving Flask app "main" (lazy loading)
* Environment: production
```

Our webapp uses a library called **Flask** to launch a local server. Let's access it!

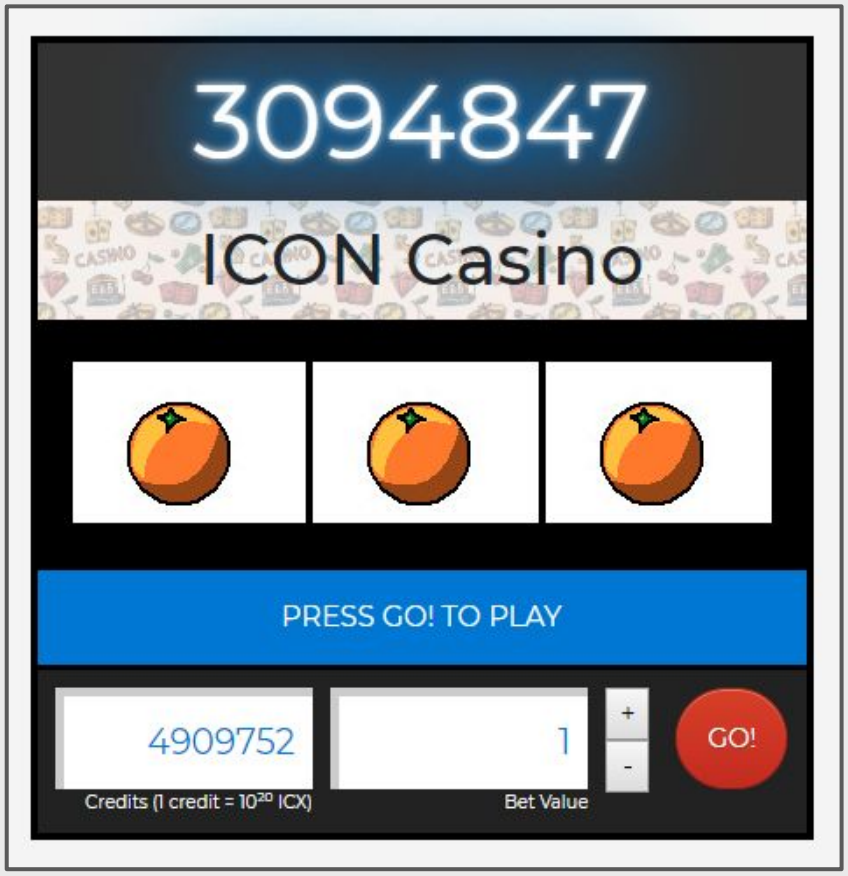
Viewing the WebApp

Head back to the dashboard tab of your browser (where you launched the workspace from).

Then click the **(i)** icon of your workspace.

Click **Retrieve** and lastly, click the **python-http** URL!

View the App



Our app is running locally - press **Go** to give it a spin!

The **Credits** box indicates how many ICX are in the application wallet.

You can modify the bet value by clicking the **+** and **-** buttons.

What's happening?

1. The user sets a bet amount and clicks **Go**.

2. The app writes a new transaction to send to our SCORE with the bet amount.

3. The SCORE receives the transaction and withdraws the bet amount from the player's wallet.

What's happening?

4 The SCORE 'spins' randomly and decides if the player has won or not.

5 If the player wins, it sends a funds transfer to the user's wallet with the winnings.

6 Lastly, it returns the result to the application so we can update the UI to the user.

Key Term

UI: User Interface. Anything that the user of an application sees on their screen or device.

Tracking the SCORE

We can track the activity of our SCORE by using the TestNet Tracker. It will provide details of all transactions on the SCORE.

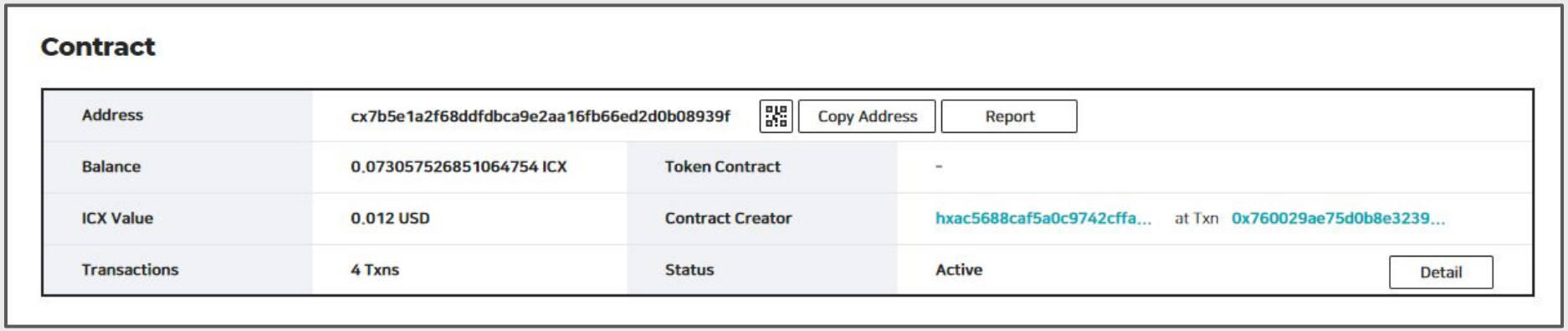
Head to the following URL:

mlhlocal.host/ICON-tracker

Paste in your **SCORE** address into the search bar and hit **Enter**.

Tracking the SCORE

Here we can view the SCORE balance, and the number of transactions that have been sent to it.



Select a transaction and you can see which wallet it came from, and how many ICX were transferred. Awesome!

Let’s recap!

We launched the app and learned how to track our SCOREs on the ICON Tracker.

Table of Contents

Let's Recap

What did you learn?

1. An Introduction to Blockchain technology and ICON
2. How to create an ICON Blockchain using Morpheus Labs
3. How to write SCOREs and deploy them using T-Bears

ICON Community Grant Program

ICON offers ICX grants to projects that benefit the ICON community!

Recommended proposal types include infrastructure, development, education, marketing, public relations, and community activities, and more! **For more info:**

http://mlhlocal.host/icongrants

What did you learn today?

We created a fun quiz to test your knowledge and see what you learned from this workshop.

http://mlhlocal.host/quiz

Learning shouldn't stop when the workshop ends...

Check your email for access to:

- These workshop slides
- Practice problems to keep learning
- Deeper dives into key topics
- Instructions to join the community
- More opportunities from MLH!


MLH localhost
Workshop
Blockchain Casino
Learning how to
SCORE on ICON
MLH localhost
ICON
MLH
MAJOR LEAGUE HACKING
v.0.0.1