Dive into ICON - Tools

ICON foundation

Dive into ICON - Tools

Step 1. ICON tools

Step 2. SCORE Samples

Step 3. Exercise

Step 4. SCORE Integration Test

Dive into ICON - Tools

Step 1. ICON tools

1. T-Bears
2. ICON SDK
3. Tracker
4. ICONex

Step 2. SCORE Samples

Step 3. Exercise

Step 4. SCORE Integration Test

Dive into ICON - Tools

Step 1. ICON tools

Step 2. SCORE Samples

1. hello_world
2. irc2_token
3. crowdsale

Step 3. Exercise

Step 4. SCORE Integration Test

Dive into ICON - Tools

Step 1. ICON tools

Step 2. SCORE Samples

Step 3. Exercise

1. Deploy hello_world
2. T-Bears
3. ICON SDK
4. ICONex

Step 4. SCORE Integration Test

Dive into ICON - Tools

Step 1. ICON tools

Step 2. SCORE Samples

Step 3. Exercise

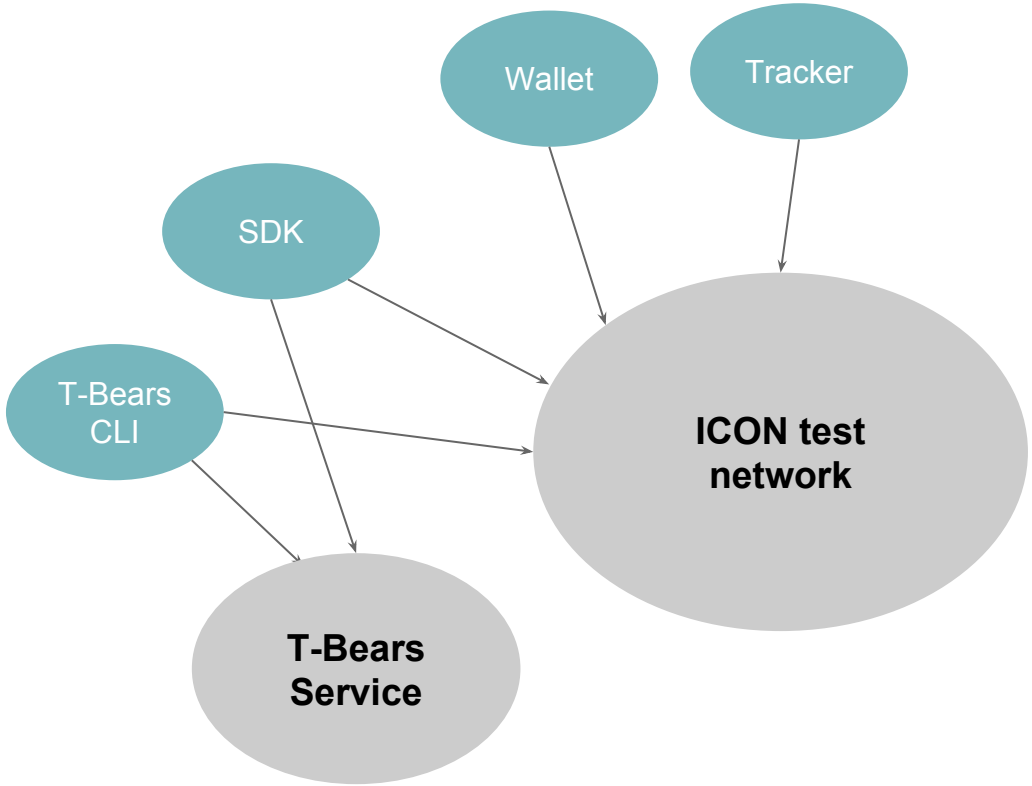
Step 4. SCORE Integration Test

1. How to Test SCORE
2. Exercise - Integration Test

1. ICON Tools

ICON Tools

- T-Bears Dev Suite
  - SCORE library
  - Service (Node emulator)
  - Test framework
  - CLI
- Client SDK
  - Java, Python, JavaScript
- ICONex
- Tracker



1.1 T-Bears

Dive into ICON - Tools

Step 1. ICON tools

1. T-Bears
2. ICON SDK
3. Tracker
4. ICONex

Step 2. SCORE Samples

Step 3. Exercise

Step 4. SCORE Integration Test

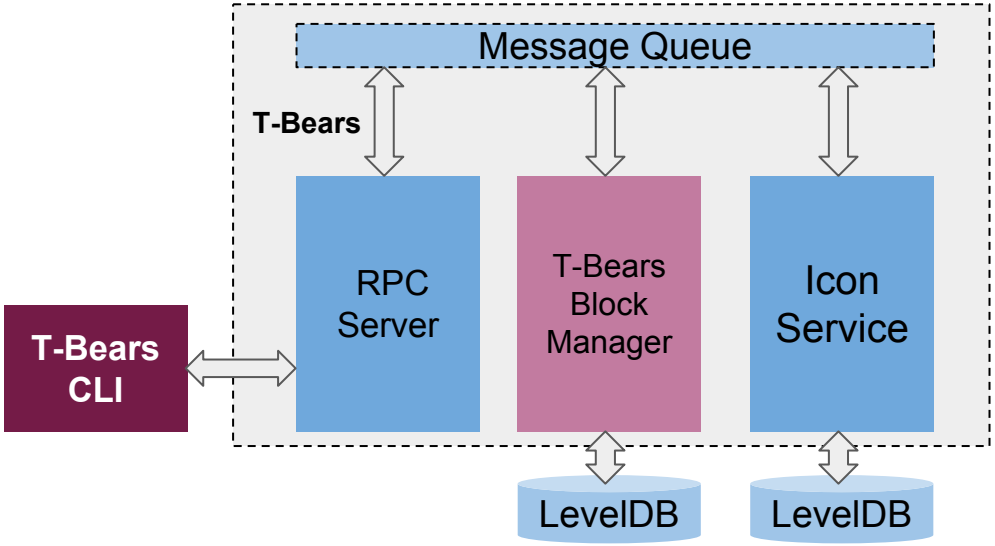
What is T-Bears?

- Definition : Test suite for ICON SCORE development

- Feature:
  - Emulated node environment.
  - Mimic block generation without consensus.
  - Generate Sample SCORE for test.
  - Transaction fee : off
  - SCORE audit : off

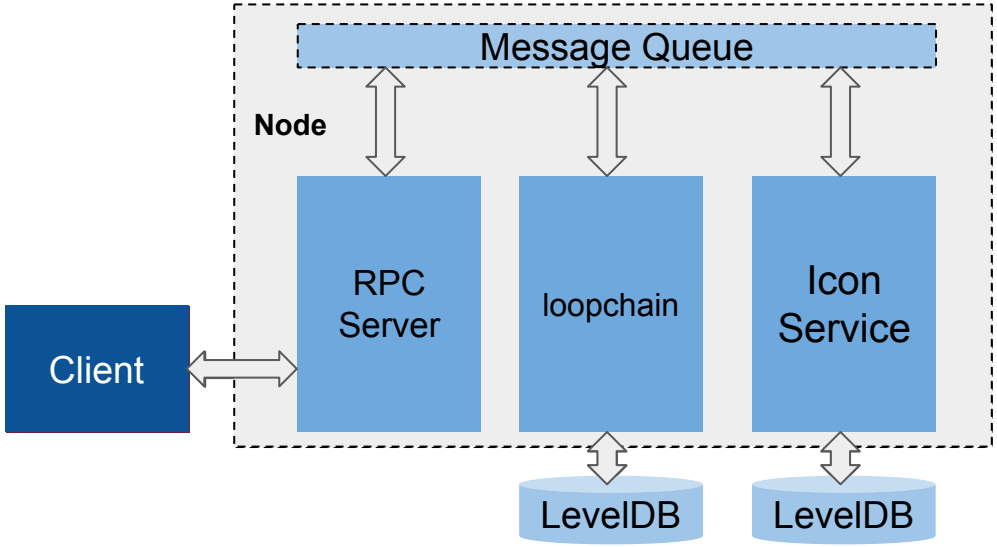
Component

- ICON RPC Server
- ICON Service
- T-Bears Block Manager
- Message queue



Difference between T-Bears and ICON Node

- Loopchain → T-Bears Block Manager
- Client → T-Bears CLI
- Node → T-Bears



Things to Concern

- Make sure three things before sending a transaction to node env.
  - Timestamp:

    Timestamp means transaction creation time in micro seconds.

    The gap between block creation time & transaction timestamp value must be within 5 minutes.

    Allowed boundary can be changed. Make sure the value on Github repo : Link

    `‘loopchain/configure_default.ALLOW_TIMESTAMP_BOUNDARY_SECOND’`

Things to Concern

- Make sure three things before sending a transaction to node env.
  - Signature:

    Signature should be included in transaction data. Except querying.

    Signature can be generated using Keystore file.

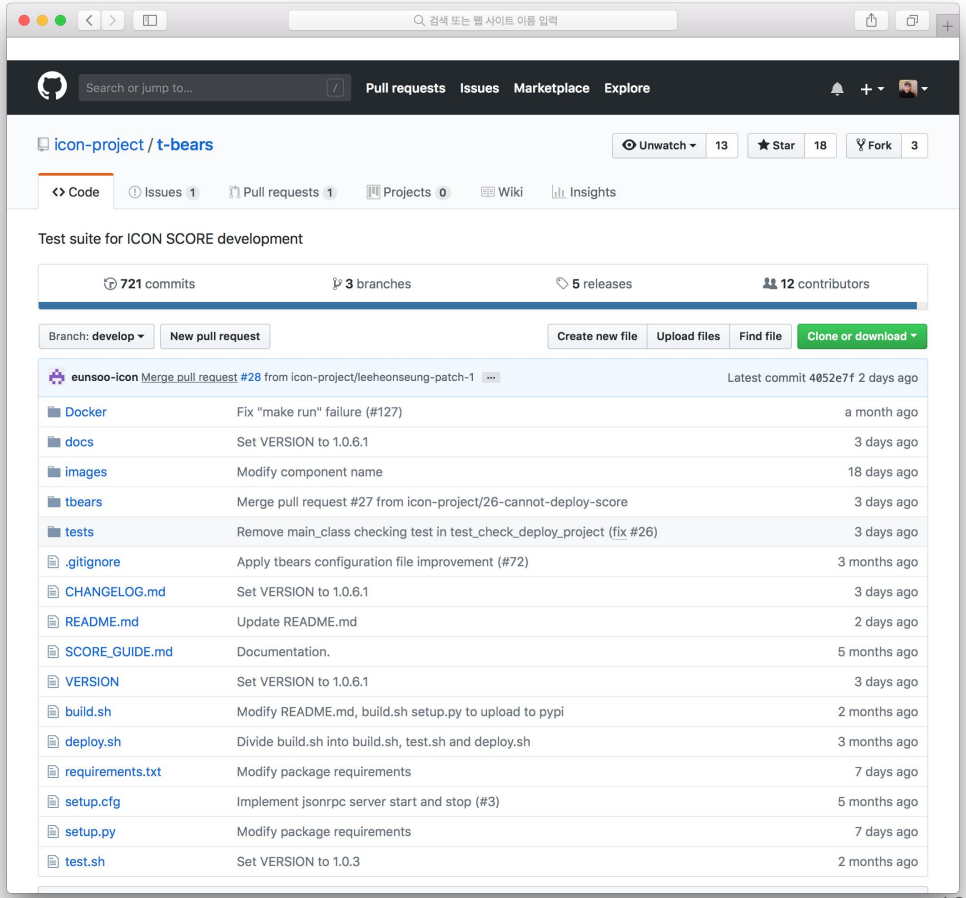
  - Steplimit:

    In case of deployment, process requires at least 10 ICX as fee.

    Make sure the estimated stepcost and set the steplimit carefully not to exceeds the balance of account.

Installing T-Bears

- T-Bears
  - Native installation
    - Git clone
    - pip install
  - Docker (Recommended)



Installing T-Bears

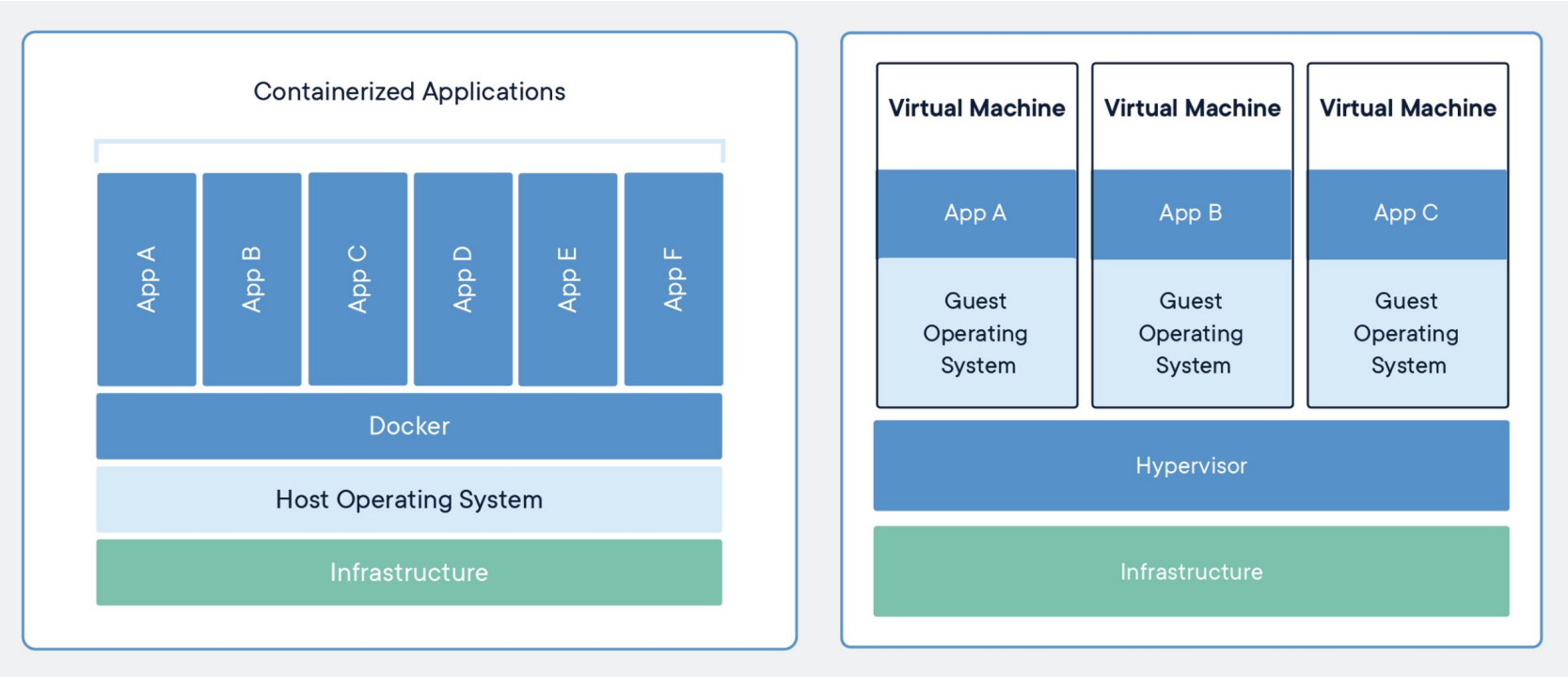
- Prerequisite
  - Install Python 3.6 version
  - Install requirements (Leveldb, autoconf, automake, libtool, pkg-config, rabbitmq)

- Install using pip
  - Activate virtualenv
  - Install tbears using pip

- Build and Install from source
  - Git clone https://github.com/icon-project/t-bears
  - Activate virtualenv
  - Build by build.sh
  - Install *.whl (pip install tbears-1.*-py3-none-any.whl)

Installing T-Bears with Docker

Why Docker?



Installing T-Bears with Docker

```
$ cd ~ && mkdir icon-workshop && cd icon-workshop
```

To install T-Bears with docker.

```
$ docker run -it --name local-tbears -p 9000:9000 -v
${PWD}:/tbears/icon-workshop iconloop/tbears
```

When it finished successfully.
You can see the activated container env.

```
$ root@07dfee84208e:/tbears#
```

Installing T-Bears with Docker

To exit Docker container.

```
$ root@07dfee84208e:/tbears# exit
```

To make your container run.

```
$ docker container start local-tbears
$ docker container attach local-tbears
```

Installing T-Bears with Docker

To escape from Docker container without terminating it.

```
Press Control + P + Q
```

To get in docker container env.

```
$ docker attach local-tbears
            or
$ docker exec -it local-tbears bash
```

T-Bears Commands

No	Command	Description
1	start	Start T-Bears service
2	stop	Stop T-Bears service
3	deploy	Deploy SCORE
4	clear	Clear all SCOREs deployed on T-Bears service
5	init	Initialize T-Bears project
6	test	Run the unittest in the SCORE

T-Bears Commands

No	Command	Description
7	genconfig	Generate T-Bears config files.(tbears_cli_config.json ,tbears_cli_config.json and keystore_test1)
8	txresult	Get transaction result by hash
9	keystore	Create a keystore file
10	balance	Get balance of given address.
11	totalsupply	Query total supply of ICX.
12	scoreapi	Get SCORE's API using given SCORE address.

T-Bears Commands

No	Command	Description
13	txbyhash	Get transaction by transaction hash.
14	lastblock	Get last block info.
15	blockbyhash	Get block info using given block hash.
16	blockbyheight	Get block info using given block height.
17	sendtx	Request icx_sendTransaction with user input json file.
18	call	Request icx_call with user input json file.

Basic Commands

No	Command	Description
1	init	Initialize T-Bears project
2	deploy	Deploy SCORE
3	txresult	Get transaction result by hash
4	call	Request icx_call with user input json file.
5	sendtx	Request icx_sendTransaction with user input json file.
6	scoreapi	Get SCORE's API using given SCORE address.

Summary

- Feature:
  - Emulated node environment.
  - Mimic block generation without consensus.
  - Generate Sample SCORE for test.
  - Transaction fee : off
  - SCORE audit : off

Summary

- Definition : Test suite for ICON SCORE development

- Feature:
  - Emulated node environment.
  - Mimic block generation without consensus.
  - Generate Sample SCORE for test.
  - Transaction fee : off
  - SCORE audit : off

Summary

No	Command	Description
1	init	Initialize T-Bears project
2	deploy	Deploy SCORE
3	txresult	Get transaction result by hash
4	call	Request icx_call with user input json file.
5	sendtx	Request icx_sendTransaction with user input json file.
6	scoreapi	Get SCORE's API using given SCORE address.

1.2 ICON SDK

Dive into ICON - Tools

Step 1. ICON tools

1. T-Bears
2. ICON SDK
3. Tracker
4. ICONex

Step 2. SCORE Samples

Step 3. Exercise

Step 4. SCORE Integration Test

ICON SDK

- SDK
  - Java [https://github.com/icon-project/icon-sdk-java]
  - Python [https://github.com/icon-project/icon-sdk-python]
  - Javascript [https://github.com/icon-project/icon-sdk-js]
- Interacting with T-Bears & ICON nodes is available using your prefered language (Java/Python/JS)

Install SDK

- Java SDK
  - Dependency management tool
    - Maven
    - Gradle
- Python SDK
  - pip install
- Javascript SDK
  - CDN
  - npm install

Install SDK

- Java SDK

1. Maven

```
<dependencies>
    <dependency>
        <groupId>foundation.icon</groupId>
        <artifactId>icon-sdk</artifactId>
        <version>[x.y.z]</version>
    </dependency>
</dependencies>
```

2. Gradle

```
dependencies { implementation 'foundation.icon:icon-sdk:[x.y.z]' }
```

Install SDK

- Python SDK (only python version 3.6)

  1. pip install

```
$ pip install iconsdk
```

- Javascript SDK

  1. CDN

```
<script src="https://cdn.jsdelivr.net/gh/icon-project/icon-sdk-js/build/icon-sdk-js.min.js"></script>
```

  2. npm install

```
$ npm install --save icon-sdk-js
```

Summary

- Official Documents
  - Java [https://github.com/icon-project/icon-sdk-java]
  - Python [https://github.com/icon-project/icon-sdk-python]
  - Javascript [https://github.com/icon-project/icon-sdk-js]

1.3 Tracker

Dive into ICON - Tools

Step 1. ICON tools

1. T-Bears
2. ICON SDK
3. Tracker
4. ICONex

Step 2. SCORE Samples

Step 3. Exercise

Step 4. SCORE Integration Test

Tracker

Through tracker, you can explore blocks, transactions, and the contracts deployed on the network. View and download contract's source code.

- Mainnet Tracker : https://tracker.icon.foundation
- Testnet Tracker : https://bicon.tracker.solidwallet.io
  - Block explorer
  - TX explorer
  - Contract explorer

Tracker - Contract List

icon Tracker

Address ▾ | Block | Transaction | Token ▾ | Address, TxHash, Block, SCORE

Addresses

Contracts

Contracts

Search for contract name / address

With verified source codes only

A total of 30 verified contract source code(s) found

Address	Contract Name	Balance	TxCount	Status ▾	Confirmed date
cx9ab3078e72c8d9017194d17b34b1a4...	AC3	0.0000 ICX	38	Active	2018-12-24
cxa3512fb64b8808ca07a8469e391b33...	Andrew Token	0.0000 ICX	2	Active	2018-11-09
cxf9148db4f8ec78823a50cb06c4fed83...	MECA Coin	0.0000 ICX	9,671	Active	2018-12-21
cx8c43c6980d57aee6a0c69fb3a2f3c30...	PATRICK	0.0000 ICX	2	Active	2018-11-23
cx3ec2814520c0096715159b8fc55fa1f...	SportToken	0.0000 ICX	10	Active	2018-12-04
cxc248ee72f58f7ec0e9a382379d67399...	WITHU	0.0000 ICX	23	Active	2018-12-05
cxf05884546c2e914db0ba1726f71598...	WithU	0.0000 ICX	2	Active	2018-12-28
cx921205acb7c51e16d5b7cbc37539a4...	weBloc	0.0000 ICX	10	Active	2018-11-09
cxb05d91b763505fcfafa082eb1488a27...	-	0.0000 ICX	2	Rejected	2018-11-09
cxbea328b67f70e23a1f05d081627224...	-	0.0000 ICX	2	Rejected	2018-11-09

Tracker - Contract Detail

icon Tracker | Address | Block | Transaction | Token | Address, TxHash, Block, SCORE

Contract

Address	cx9ab3078e72c8d9017194d17b34b1a47b661945ca	Copy Address	
Balance	0 ICX	Token Contract	AC3 (AC3)
ICX Value	0.000 USD	Contract Creator	hxa702a40b4089c7fb6d9a... at Txn 0xb89690b7598e07c286d...
Transactions	38 Txns	Status	Active Detail

Transactions | Internal Transactions | Token Transfers | **Code** | Read Contract | Events

Contract Name	Active Contract Source Code	Updated Contract Source Code
AC3 (AC3)	Download	-

Contract ABI Copy ABI

```
[
    {
        "type": "function",
        "name": "balanceOf",
        "inputs": [
```

Tracker - Block List

icon Tracker | Address | Block | Transaction | Token | Address, TxHash, Block, SCORE

Blocks

A total of 157,740 total block(s) found (Showing the last 157,740 record(s) only)

Block	Age	No of Txns	Block Hash	Amount	TxFee
157,739	38 seconds ago	1	0xaec14d02b1a86eececbbd782946a5a62d53d9cd8f8311240e0756c8e1caa...	0.0020 ICX	0.0010 ICX
157,738	2 minutes ago	1	0x022d709191dc0ee8273f3b02321546e1663018b7474025a9174b3e71663...	20.8890 ICX	0.0010 ICX
157,737	4 minutes ago	1	0x002e43c7f682007ac74890e05ecac1030b090ca16a5412579f3995178449...	0.0030 ICX	0.0010 ICX
157,736	7 minutes ago	1	0x2bf2d728bbd4efc61de8ff7ad88823dd89d2f2b517daea58b0bb281f5d5d5...	0.0040 ICX	0.0010 ICX
157,735	10 minutes ago	1	0x9f157bd4bac2266ab4142fa585ab01a876deb5174b2bfdea16ad94e8de7af...	0.0010 ICX	0.0010 ICX
157,734	12 minutes ago	1	0x2399bdd60287ef42db6aea3fa7f313926e5373fe808e68e12e18cf46bbc4a...	7.2467 ICX	0.0010 ICX
157,733	12 minutes ago	1	0x0fddaedc7b44c8da3bf0bc80c1986236de263199a4fbe1ddd632218f1c702...	1,710.9173 ICX	0.0010 ICX
157,732	12 minutes ago	1	0xf49a824c4f336cddf9628a7f8eff7c619d00e6ee0170dfdaf507cfe162f49bcf	0.9800 ICX	0.0010 ICX
157,731	13 minutes ago	1	0x23dcba546cd1185804c7d199abd115bef46e0ed072e545cbb9d10d8ac1c2...	0.0010 ICX	0.0010 ICX
157,730	16 minutes ago	1	0x8489bbd775e6a69aab00b1c76b641d313fe5435475c46a3ecda42d7052fa...	0.0050 ICX	0.0010 ICX

Tracker - Transaction List

icon Tracker | Address | Block | Transaction | Token | Address, TxHash, Block, SCORE

Transactions

A total of 163,323 total transaction(s) found (Showing the last 163,323 record(s) only)

TxHash	Block	Age	From		To	Amount	TxFee
0x2cb4344726d93e276e9c1f02...	157,739	3 minutes ago	hx522bff55a62e...	→	hx11de4e28be48...	0.0020 ICX	0.0010 ICX
0xe938ac1f60df242ceb6979549...	157,738	4 minutes ago	hxab2cee960e38...	→	hx5130e850466...	20.8890 ICX	0.0010 ICX
0xffbbd164d5afda11af308db7d...	157,737	6 minutes ago	hx522bff55a62e...	→	hx11de4e28be48...	0.0030 ICX	0.0010 ICX
0x24932fbba95d335f14e84629...	157,736	9 minutes ago	hx522bff55a62e...	→	hx11de4e28be48...	0.0040 ICX	0.0010 ICX
0x522036587192b1579ace9d17...	157,735	12 minutes ago	hx522bff55a62e...	→	hx11de4e28be48...	0.0010 ICX	0.0010 ICX
0x97d287bc5ddef620f801cf035...	157,734	15 minutes ago	hxc9dbb5d27c15...	→	hxe8c78d8dd256...	7.2467 ICX	0.0010 ICX
0x6a3e84dc1212c5cd6465a2a5...	157,733	14 minutes ago	hxc4193cda4a75...	→	hx8e72de3bc0ca...	1,710.9173 ICX	0.0010 ICX
0xf19d6c10edbd7810834f1a1cf...	157,732	14 minutes ago	hxc4193cda4a75...	→	hx186e4c79352d...	0.9800 ICX	0.0010 ICX
0xe240d592074f964f7361a514...	157,731	15 minutes ago	hx522bff55a62e...	→	hx11de4e28be48...	0.0010 ICX	0.0010 ICX
0xa9e89fe05c05d91824029b8ca...	157,730	18 minutes ago	hx522bff55a62e...	→	hx11de4e28be48...	0.0050 ICX	0.0010 ICX

Tracker - Transaction Detail

icon Tracker | Address | Block | Transaction | Token | Address, TxHash, Block, SCORE

Transaction

TxHash	0x107fd55f1e932e13988c33e911bf3100e7fd45c79bc3df9fbedf46b0666796f4 Copy TxHash
Status	Success
Block Height	157,741 (1 Confirmation(s))
Time Stamp	2019-01-02 14:33:31 (UTC+9, 1 minute ago)
From	hxc4193cda4a75526bf50896ec242d6713bb6b02a3 Copy Address
To	hx186e4c79352db3d8e5e47b88e8ceb62061f147fd Copy Address
Amount	498.51 ICX
Step Limit	100,000
Step used by Txn	100,000
Step Price	0.00000001 ICX (10 Gloop)
Actual TxFee	0.001 ICX (0.000 USD)

Tracker - IRC2 Tokens

ICON Tracker | Address | Block | Transaction | Token | Address, TxHash, Block, SCORE

Tokens
Token Transfers

Tokens

Search for any token name / address / symbol

ICON Tokens Market Capitalization Sorted by MarketCap value in DESC Order

A total of 8 IRC2 Token Contract(s) found (Sorted by MarketCap value in DESC Order)

No.	Token	Price	% Change	Volume (24h)	MarketCap
1	AC3 (AC3)	- USD - ICX - BTC - ETH	- %	- USD	- USD
2	Andrew Token (ATK)	- USD - ICX - BTC - ETH	- %	- USD	- USD
3	MECA Coin (MCA)	- USD - ICX - BTC - ETH	- %	- USD	- USD
4	PATRICK (PAT)	- USD - ICX - BTC - ETH	- %	- USD	- USD
5	SportToken (SPORT)	- USD - ICX - BTC - ETH	- %	- USD	- USD
6	WITHU (WTU)	- USD - ICX - BTC - ETH	- %	- USD	- USD
7	WithU (WTU)	- USD - ICX - BTC - ETH	- %	- USD	- USD

1.4. ICONex

Dive into ICON - Tools

Step 1. ICON tools

1. T-Bears
2. ICON SDK
3. Tracker
4. ICONex

Step 2. SCORE Samples

Step 3. Exercise

Step 4. SCORE Integration Test

ICONex

ICONex is chrome extension wallet for ICON.

ICONex's features

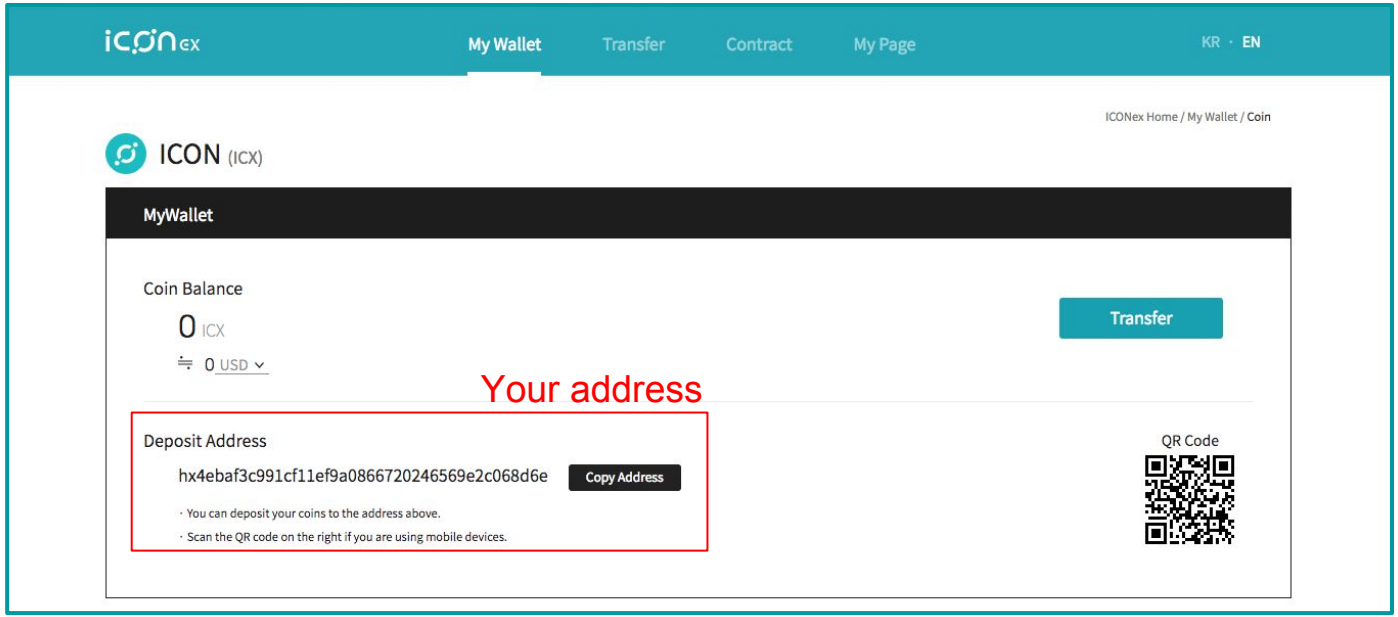
- Send transaction
- Create account
  - Account address prefix 'hx' for EOA, 'cx' for CA
- Make query call
- …

1. Install ICONex & Create an account

- Install ICONex from the chrome web store : https://chrome.google.com/webstore/detail/iconex/flpiciilemghbmfalicajoolhkkenfel?hl=en

- Create an account : https://icon-project.github.io/docs/wallet.html#using-iconex

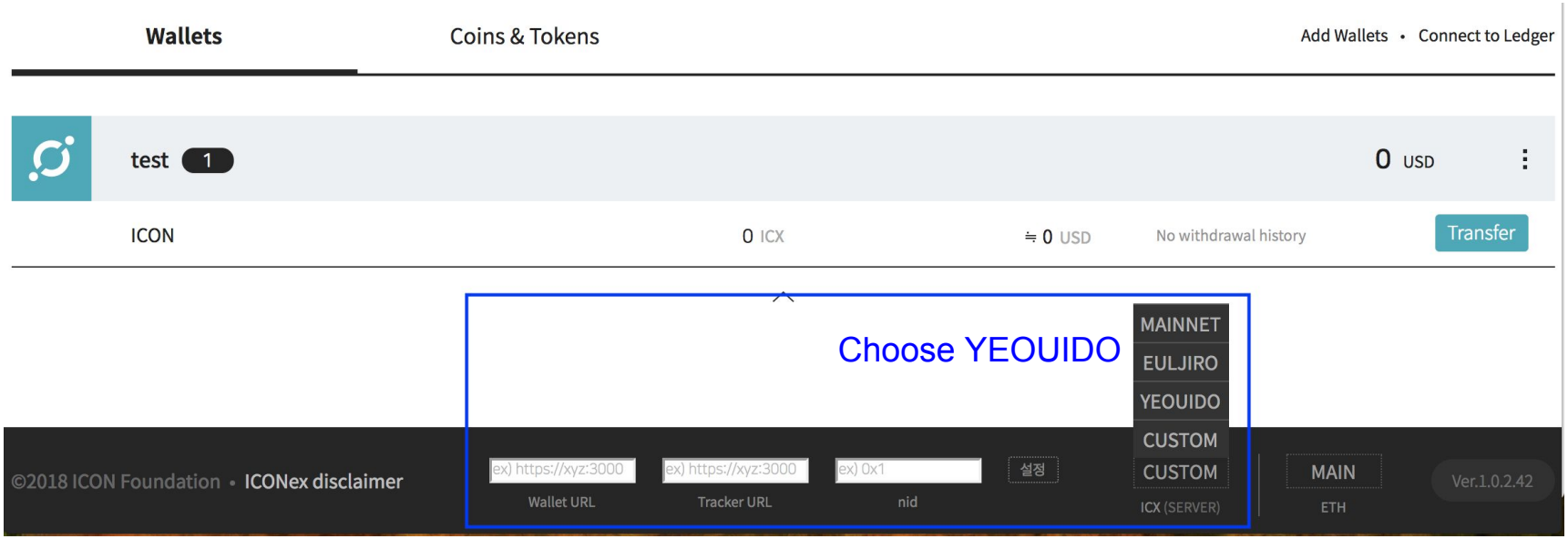
- Youtube video guide : https://www.youtube.com/watch?v=9A1XHLNbsQk



- EOA (External Owned Account) : eg) **hx**e7af5fcfd8dfc67530a01a0e403882687528dfcb
- CA (Contract Account) : eg) **cx**3eaf5fcfd5dfc67520a01a0e403682687526afca

2. Switch to the testnet

- Guide : https://icon-project.github.io/docs/icon_network.html#how-to-change-network-in-iconex-chrome-extension



2. SCORE Samples

SCORE Samples

- ICON provides `hello_world`, `irc2_token`, `crowdsale` as samples
- A sample has following structure.

```
sample
├──my_score
  ├── __init__.py
  ├── my_score.py
  ├── package.json
  ├── ...
├──tests
├──README.md
├──deploy.json
```

2.1 hello_world

Dive into ICON - Tools

Step 1. ICON tools

Step 2. SCORE Samples

1. hello_world
2. irc2_token
3. crowdsale

Step 3. Exercise

Step 4. SCORE Integration Test

hello_world

- The simplest SCORE that returns "Hello" in the response message.

```
hello_world
├──hello_world
  ├── __init__.py
  ├── hello_world.py
  ├── package.json
├──tests
├──README.md
```

hello_world

- IconScoreBase
  - Abstract methods
    - __init__
    - on_install
    - on_update

hello_world

- HelloWorld
  - Methods
    - name: @external(readonly=True).
    - hello: @external(readonly=True).

2.2 irc2_token

Dive into ICON - Tools

Step 1. ICON tools

Step 2. SCORE Samples

1. hello_world
2. irc2_token
3. crowdsale

Step 3. Exercise

Step 4. SCORE Integration Test

irc2_token

- Provide a simple and solid reference implementation for ICON token standard (IRC2)

```
irc2_token
├──sample_token
  ├── __init__.py
  ├── sample_token.py
  ├── package.json
  ├──tests
├──README.md
├──sampletoken.json
```

What is Token?

- ERC20, ERC223
  - Ethereum token technical standard interface.

- IRC2
  - A standard interface for tokens on ICON network.

irc2_token

- SampleToken
  - Methods
    - name: @external(readonly=True)
    - symbol: @external(readonly=True)
    - decimals: @external(readonly=True)
    - totalSupply: @external(readonly=True)

irc2_token

- SampleToken
  - Methods
    - balanceOf: @external(readonly=True)
    - Transfer: @eventlog(indexed=3)
    - transfer: @external
    - _transfer

irc2_token

- InterfaceScore
  - TokenFallbackInterface
    - tokenFallback: @interface

2.3 crowdsale

Dive into ICON - Tools

Step 1. ICON tools

Step 2. SCORE Samples

1. hello_world
2. irc2_token
3. crowdsale

Step 3. Exercise

Step 4. SCORE Integration Test

crowdsale

- Provide a sample implementation of crowdsale.

```
crowdsale
├──sample_crowdsale
  ├── __init__.py
  ├── sample_crowdsale.py
  ├── package.json
  ├──tests
├──README.md
├──crowdsale.json
```

What is crowdsale?

- crowdsale
  - A type of crowdfunding that issues cryptocurrency tokens that are stored on the user's device.

- Selling token to crowd
  - Goal
  - D-Day

crowdsale

- SampleCrowdSale
  - Methods
    - FundTransfer: @eventlog(indexed=3)
    - GoalReached: @eventlog(indexed=2)
    - tokenFallback: @external
    - fallback: @payable

crowdsale

- SampleCrowdSale
  - Methods
    - totalJoinerCount: @external(readoly=True)
    - _after_dead_line
    - checkGoalReached: @external
    - safeWithdrawal: @external

crowdsale

- InterfaceScore
  - TokenInterface
    - transfer: @interface

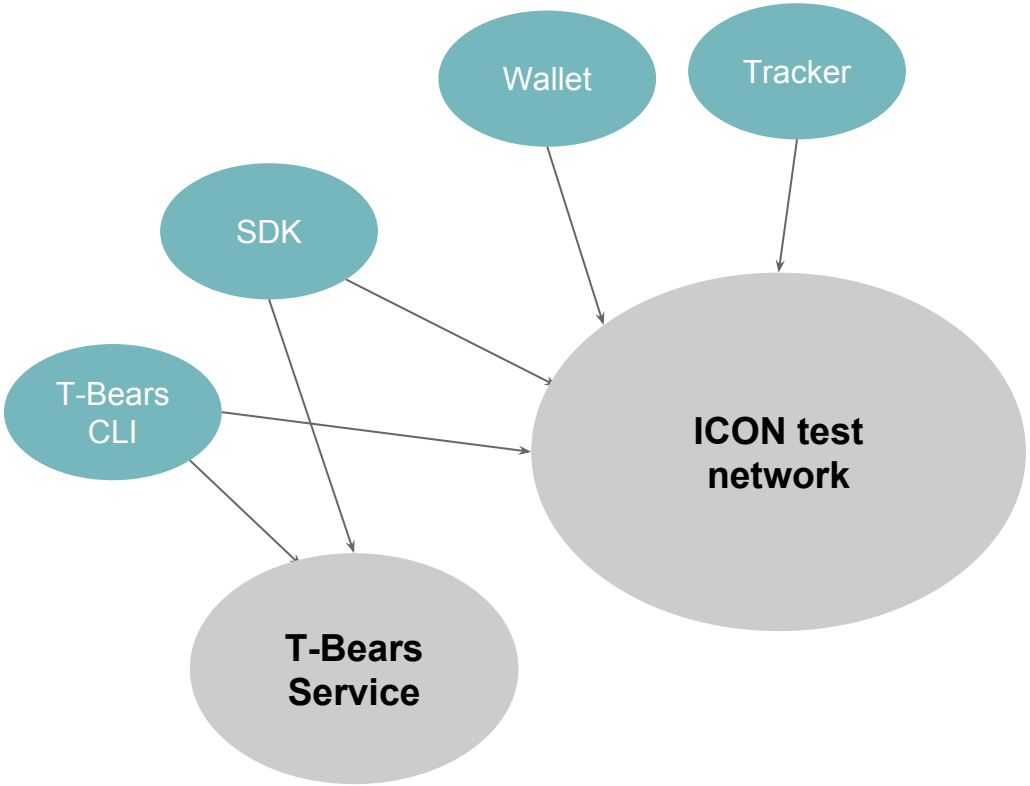
Summary

- Sample SCORE
  - hello_world : The simplest yet completely legitimate SCORE that returns "Hello" in the response message.
  - irc2_token : Provide a simple and solid reference implementation for ICON token standard (IRC2)
  - crowdsale : Provide a sample implementation of crowdsale.

3. Exercise

Exercise

- T-Bears Dev Suite
  - SCORE library
  - Service (Node emulator)
  - Test framework
  - CLI
- Client SDK
  - Java, Python, JavaScript
- ICONex
- Tracker



3.1 Deploy hello_world

Dive into ICON - Tools

Step 1. ICON tools

Step 2. SCORE Samples

Step 3. Exercise

1. Deploy hello_world
2. T-Bears
3. ICON SDK
4. ICONex

Step 4. SCORE Integration Test

Deploy hello_world - YEOUIDO

No	Command	Description
1	init	Initialize T-Bears project
2	deploy	Deploy SCORE
3	txresult	Get transaction result by hash
4	call	Request icx_call with user input json file.
5	sendtx	Request icx_sendTransaction with user input json file.
6	scoreapi	Get SCORE's API using given SCORE address.

Deploy hello_world - YEOUIDO

- ***tbears init hello_world*** : Initialize the SCORE project named hello_world

```
hello_world
├──hello_world
  ├── __init__.py
  ├── hello_world.py
  ├── package.json
├──tests
├──README.md
```

Deploy hello_world - YEOUIDO

- ***tbears deploy hello_world -k <key> -c deploy.json***

```
{
    "uri": "https://bicon.net.solidwallet.io/api/v3",
    "nid": "0x3",
    "to": "cx0000000000000000000000000000000000000000",
    "deploy": {
        "stepLimit": "0x77359400",
        "mode": "install",
        "scoreParams": {}
    },
    "txresult": {},
    "transfer": {
        "stepLimit": "0x77359400"
    }
}
```

3.2 T-Bears

Dive into ICON - Tools

Step 1. ICON tools

Step 2. SCORE Samples

Step 3. Exercise

1. Deploy hello_world
2. T-Bears
3. ICON SDK
4. ICONex

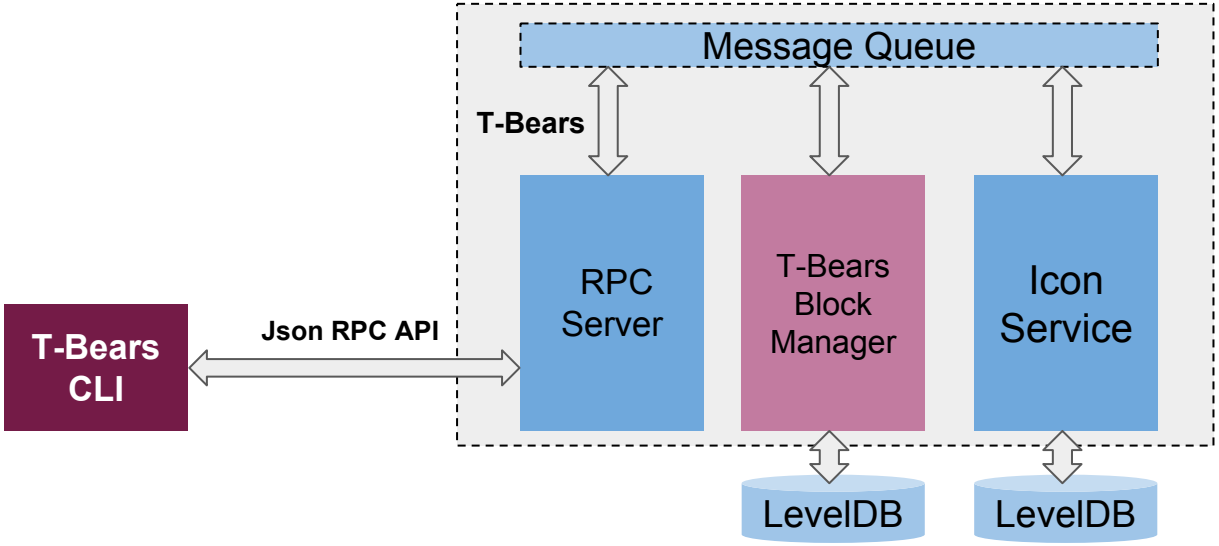
Step 4. SCORE Integration Test

Basic Commands

No	Command	Description
1	init	Initialize T-Bears project
2	deploy	Deploy SCORE
3	txresult	Get transaction result by hash
4	call	Request icx_call with user input json file.
5	sendtx	Request icx_sendTransaction with user input json file.
6	scoreapi	Get SCORE's API using given SCORE address.

How does T-Bears CLI work with RPC Server?

- T-Bears CLI interacts with RPC Server using JSON RPC API protocol



ICON JSON-RPC API V3

- icx_sendTransaction
  - Transfer designated amount of ICX coins from 'from' address to 'to' address.
  - Install a new SCORE.
  - Update the SCORE in the 'to' address.
  - Invoke a function of the SCORE in the 'to' address.
  - Transfer a message.

ICON JSON-RPC API V3

- icx_sendTransaction
  - Transfer designated amount of ICX coins from 'from' address to 'to' address.


```
{
    "jsonrpc": "2.0",
    "method": "icx_sendTransaction",
    "id": 1234,
    "params": {
        "version": "0x3",
        "from": "hxbe258ceb872e08851f1f59694dac2558708ece11",
        "to": "hx5bfdb090f43a808005ffc27c25b213145e80b7cd",
        "value": "0xde0b6b3a7640000",
        "stepLimit": "0x12345",
        "timestamp": "0x563a6cf330136",
        "nid": "0x3",
        "nonce": "0x1",
        "signature":"VAia7YZ2Ji6igKWzjR2YsGa2m53nKPrfK7uXYW78QLE+ATehAVZPC40szvAiA6NEU5gCYB4c4qaQzqDh2ugcHgA=”
    }
}
```

ICON JSON-RPC API V3

- icx_sendTransaction
  - Install a new SCORE.

```
...
"params": {
    ...
    "dataType": "deploy",
    "data": {
        "contentType": "application/zip",
        "content": "0x1867291283973610982301923812873419826abcdef91827319263187263a7326e...",
        "params": {
            "name": "ABCToken"
        }
    }
}
...
```

ICON JSON-RPC API V3

- icx_sendTransaction
  - Invoke a function of the SCORE in the 'to' address.

```
...
"params": {
    ...
    "dataType": "call",
    "data": {
        "method": "transfer",
        "params": {
            "to":"hxab2d8215eab14bc6bdd8bfb2c8151257032ecd8b",
            "value": "0x1"
        }
    }
}
...
```

ICON JSON-RPC API V3

- icx_call
  - Calls SCORE's external function. Does not make state transition (i.e., read-only).

```
{
    "jsonrpc": "2.0",
    "method": "icx_call",
    "id": 1234,
    "params": {
        "from": "hxbe258ceb872e08851f1f59694dac2558708ece11",
        "to": "cxb0776ee37f5b45bfaea8cff1d8232fbb6122ec32",
        "dataType": "call",
        "data": {
            "method": "get_balance",
            "params": {
                "address": "hx1f9a3310f60a03934b917509c86442db703cbd52"
            }
        }
    }
}
```

ICON JSON-RPC API V3

- JSON-RPC also provides various APIs like
  - icx_getLastBlock
  - icx_getBlockByHeight
  - icx_getBlockByHash
  - icx_getBalance
  - icx_getScoreApi
  - icx_getTotalSupply
  - icx_getTransactionResult
  - icx_getTransactionByHash

Deploy SCORE with T-Bears

- ***tbears deploy -k <key> -c deploy.json <scoreproject>***

```
{
  "uri": "https://bicon.net.solidwallet.io/api/v3",
  "nid": "0x3",
  "to": "cx0000000000000000000000000000000000000000",
  "deploy": {
    "mode": "install",
    "scoreParams": {
      "initialSupply":"1000",
      "decimals":"10"
    }
  },
  "txresult": {},
  "transfer": {
    "stepLimit": "0x77359400"
  }
}
```

Check Tx Result & SCORE API with T-Bears

- ***tbears txresult <txhash>:***
  with -u https://bicon.net.solidwallet.io/api/v3
  txhash can be derived from response of executed deploy request.
  You can select the network to send request with -u option

- ***tbears scoreapi <scoreAddress>:***
  with -u https://bicon.net.solidwallet.io/api/v3
  scoreAddress can be derived from response of executed txresult request.
  You can select the network to send request with -u option

Send Transaction with T-Bears (Transfer)

- ***tbears transfer -c transfer.json <to_address> <icx_amount: decimal>***

```
{
    "jsonrpc": "2.0",
    “keyStore”: “./myKey”,
    "method": "icx_sendTransaction",
    "params": {
        "version": "0x3",
        "stepLimit": "0x3000000"
    },
    "id": 1
}
```

Send Transaction with T-Bears (Function Call)

- ***tbears sendtx -k <key> sendtx.json***

```
{
        "jsonrpc": "2.0",
        "method": "icx_sendTransaction",
        "params": {
         "version":"0x3",
         "stepLimit":"0x77359400",
         "timestamp":"0x3000000",
         "nid":"0x3",
         "to": "cxa8d205969e06ebd3373ef2ebae73b4cfe8996919",
         "dataType": "call",
         "data": {
         "method": "hello"
   }
  }
}
```

Query Call with T-Bears

- ***tbears call call.json***

```
{
  "jsonrpc": "2.0",
  "method": "icx_call",
  "params": {
    "from": "hxef73db5d0ad02eb1fadb37d0041be96bfa56d4e6",
    "to": "cx02b13428a8aef265fbaeeb37394d3ae8727f7a19",
    "dataType": "call",
    "data": {
      "method": "name"
    }
  },
  "id": 1
}
```

3.3 ICON SDK

Dive into ICON - Tools

Step 1. ICON tools

Step 2. SCORE Samples

Step 3. Exercise

1. Deploy hello_world
2. T-Bears
3. ICON SDK
4. ICONex

Step 4. SCORE Integration Test

Deploy SCORE with PythonSDK

- IconService & KeyWallet

```
from iconsdk.icon_service import IconService
from iconsdk.providers.http_provider import HTTPProvider
from iconsdk.wallet.wallet import KeyWallet

# Creates an IconService instance using the HTTP provider and set a provider.
icon_service = IconService(HTTPProvider("https://iconx.io"))

# Loads a wallet from bytes of the private key
wallet = KeyWallet.load(b'-B\x99...xedy')

# Loads a wallet from a keystore file
wallet = KeyWallet.load("./keystore", "password")
```

Deploy SCORE with PythonSDK

- DeployTransactionBuilder

```
from iconsdk.builder.transaction_builder import DeployTransactionBuilder

# Generates an instance of transaction for deploying SCORE.
transaction = DeployTransactionBuilder()\
    .from_(wallet.get_address())\
    .to("cx00...02")\
    .step_limit(1_000_000)\
    .nid(3)\
    .nonce(100)\
    .content_type("application/zip")\
    .content(b'D8\xe9...\xfc')\
    .params(params)\
    .build()
```

Deploy SCORE with PythonSDK

- SignedTransaction & Send transaction

```
from iconsdk.signed_transaction import SignedTransaction
from iconsdk.icon_service import IconService
from iconsdk.providers.http_provider import HTTPProvider

# Creates an IconService instance using the HTTP provider and set a provider.
icon_service = IconService(HTTPProvider("https://iconx.io"))

# Generate an instance of transaction
…

# Returns the signed transaction object having a signature
signed_transaction = SignedTransaction(transaction, wallet)

# Sends the transaction
tx_hash = icon_service.send_transaction(signed_transaction)
```

Check Transaction Result with PythonSDK

- Get transaction result

```
from iconsdk.icon_service import IconService
from iconsdk.providers.http_provider import HTTPProvider

# Creates an IconService instance using the HTTP provider and set a provider.
icon_service = IconService(HTTPProvider("https://iconx.io"))

tx_result = icon_service.get_transaction_result("0x000...000")
```

Check SCORE API with PythonSDK

- Get SCORE's API

```
from iconsdk.icon_service import IconService
from iconsdk.providers.http_provider import HTTPProvider

# Creates an IconService instance using the HTTP provider and set a provider.
icon_service = IconService(HTTPProvider("https://iconx.io"))

# Returns SCORE's external API list
score_apis = icon_service.get_score_api("cx000...1")
```

Send Transaction with PythonSDK (Transfer)

- TransactionBuilder

```
from iconsdk.builder.transaction_builder import TransactionBuilder

# Generates an instance of transaction for sending icx.
transaction = TransactionBuilder()\
    .from_(wallet.get_address())\
    .to("cx00...02")\
    .value(150000000)\
    .step_limit(1000000)\
    .nid(3)\
    .nonce(100)\
    .build()
```

Send Transaction with PythonSDK (Transfer)

- Sign & Send transaction

```
from iconsdk.signed_transaction import SignedTransaction
from iconsdk.icon_service import IconService
from iconsdk.providers.http_provider import HTTPProvider

# Creates an IconService instance using the HTTP provider and set a provider.
icon_service = IconService(HTTPProvider("https://iconx.io"))

# Generate an instance of transaction
…

# Returns the signed transaction object having a signature
signed_transaction = SignedTransaction(transaction, wallet)

# Sends the transaction
tx_hash = icon_service.send_transaction(signed_transaction)
```

Send Transaction with PythonSDK (Function Call)

- CallTransactionBuilder

```
from iconsdk.builder.transaction_builder import CallTransactionBuilder,

# Generates an instance of transaction for calling method in SCORE.
transaction = CallTransactionBuilder()\
    .from_(wallet.get_address())\
    .to("cx00...02")\
    .step_limit(1000000)\
    .nid(3)\
    .nonce(100)\
    .method("transfer")\
    .params(params)\
    .build()
```

Send Transaction with PythonSDK (Function Call)

- Sign & Send transaction

```
from iconsdk.signed_transaction import SignedTransaction
from iconsdk.icon_service import IconService
from iconsdk.providers.http_provider import HTTPProvider

# Creates an IconService instance using the HTTP provider and set a provider.
icon_service = IconService(HTTPProvider("https://iconx.io"))

# Generate an instance of transaction
…

# Returns the signed transaction object having a signature
signed_transaction = SignedTransaction(transaction, wallet)

# Sends the transaction
tx_hash = icon_service.send_transaction(signed_transaction)
```

Query Call with PythonSDK

- CallBuilder

```
from iconsdk.icon_service import IconService
from iconsdk.providers.http_provider import HTTPProvider

# Creates an IconService instance using the HTTP provider and set a provider.
icon_service = IconService(HTTPProvider("https://iconx.io"))

# Generates a call instance using the CallBuilder
call = CallBuilder()\
        .from_(wallet.get_address())\
        .to("cx000...1")\
        .method("balance_of")\
        .params({"address": "hx000...1"})\
        .build()

# Calls SCORE's external function which is read-only without creating a transaction on Loopchain
result = icon_service.call(call)
```

3.4 ICONex

Dive into ICON - Tools

Step 1. ICON tools

Step 2. SCORE Samples

Step 3. Exercise

1. Deploy hello_world
2. T-Bears
3. ICON SDK
4. ICONex

Step 4. SCORE Integration Test

ICONex

- The easiest way to interact with ICON network.
- Supports various functions

- Scenario:
  1. Check Transaction History with ICONex
  2. Check SCORE API with ICONex
  3. Query Call with ICONex
  4. Send Transaction with ICONex (Transfer)
  5. Send Transaction with ICONex (Function Call)

Summary

Interacting with ICON nodes & T-Bears

- Protocol
  - ICON JSON-RPC API V3

- Tools
  - T-Bears CLI
  - ICON SDK
  - ICONex

4. SCORE Integration Test

SCORE Integration Test

- How to Run the SCORE test
  - ***$ tbears test <score_project_path>***
  - ***$ python -m unittest discover <score_project_path>***

- Package and modules
  - ICON python SDK
  - IconIntegrateTestBase in T-Bears

4.1 How to Test SCORE

Dive into ICON - Tools

Step 1. ICON tools

Step 2. SCORE Samples

Step 3. Exercise

Step 4. SCORE Integration Test

1. How to Test SCORE
2. Exercise - Integration Test

How to write SCORE integration test code

- Must inherit IconIntegrateTestBase class
  - Support python unittest
    - You can write and run the test method with prefix ‘test_’
    - You can initialize the test by override setUp method
    - You can finalize the test by override tearDown method
  - Emulate ICON service for test
    - self._test1 : Account with 1,000,000 ICX
    - self._wallet_array[] : 10 empty accounts in list

How to write SCORE integration test code

- Must inherit IconIntegrateTestBase class
  - Provides API for SCORE integration test
    - process_transaction():

      Invoke transaction and return transactin result
    - process_call():

      Calls SCORE's external function which is read-only and return result

SCORE test code with PythonSDK (Only)

- IconService

```
Unittest
self.icon_service = None

T-Bears emulated env.
self.icon_service = IconService(HTTPProvider("http://localhost:9000/api/v3"))

YEOUIDO Testnet env.
self.icon_service = IconService(HTTPProvider("https://bicon.net.solidwallet.io/api/v3"))
```

SCORE test code with PythonSDK (Only)

- process_transaction()

```
def process_transaction(self, request: SignedTransaction,
                        network: IconService = None,
                        block_confirm_interval: int = tbears_server_config[TbConf.BLOCK_CONFIRM_INTERVAL]) -> dict:
    try:
        if network is not None:
            # Send the transaction to network
            tx_hash = network.send_transaction(request)
            sleep(block_confirm_interval)
            # Get transaction result
            tx_result = network.get_transaction_result(tx_hash)
        else:
            # process the transaction in local
            tx_result = self._process_transaction_in_local(request.signed_transaction_dict)
    except IconServiceBaseException as e:
        tx_result = e.message

    return tx_result
```

SCORE test code with PythonSDK (Only)

- process_call()

```
def process_call(self, call: Call, network: IconService = None):
    try:
        if network is not None:
            response = network.call(call)
        else:
            request = {
                "from": Address.from_string(call.from_),
                "to": Address.from_string(call.to),
                "dataType": "call",
                "data": {
                    "method": call.method
                }
            }

            if isinstance(call.params, dict):
                request["data"]["params"] = call.params

            response = self._query(request=request)
    except IconServiceBaseException as e:
        response = e.message

    return response
```

Summary

- ICON SCORE Integration Test
  - Support python unittest
  - Emulate ICON service for test
  - Provides API for SCORE integration test

4.2 Exercise - Integration Test

Dive into ICON - Tools

Step 1. ICON tools

Step 2. SCORE Samples

Step 3. Exercise

Step 4. SCORE Integration Test

1. How to Test SCORE
2. Exercise - Integration Test

Exercise - setUp & tearDown

- setUp & tearDown (Optional method)

```
def setUp(self):
      super().setUp()
      print('setup')

      self.icon_service = None
      # self.icon_service=IconService(HTTPProvider("http://localhost:9000/api/v3"))

def tearDown(self):
      print('tearDown')
```

Exercise - Create & Sign Transaction

- PythonSDK TransactionBuilder (Deploy)

```
from iconsdk.builder.transaction_builder import DeployTransactionBuilder

 # Generates an instance of transaction for deploying SCORE.
 transaction = DeployTransactionBuilder()\
    .from_(wallet.get_address())\
    .to("cx00...02")\
    .step_limit(1000000)\
    .nid(3)\
    .nonce(100)\
    .content_type("application/zip")\
    .content(b'D8\xe9...\xfc')\
    .params(params)\
    .build()

 # Returns the signed transaction object having a signature
 signed_transaction = SignedTransaction(transaction, self.test1_wallet)
...
```

Exercise - Process Transaction & Verification

- Process transaction & Verify the transaction result

```
...

tx_result = self.process_transaction(signed_transaction, self.icon_service)

self.assertTrue('status' in tx_result)
self.assertEqual(1, tx_result['status'])
self.assertTrue('scoreAddress' in tx_result)
```

Exercise - Create Query Call & Process Call

- PythonSDK Call & Process Call

```
call = CallBuilder().from_(_from.get_address()) \
       .to(self._sample_game_score_address) \
       .method("show_game_room_list") \
       .build()

# Sends the call request
response = self.process_call(call, self.icon_service)
return response
```

Summary

- ICON SCORE Integration Test Rules
  - setUp : Initialize the test with customized method
  - tearDown : Finalize the test with customized method
  - PythonSDK : Create & sign transaction for SCORE test
  - IconService : Execute transaction & query call
  - Verification : Verify the result of executed transaction

Dive into ICON - Appendix

Dive into ICON - Appendix

Appendix A. AWS Development Network

Appendix B. Development Resources

Appendix A. AWS Development Network

Dive into ICON - Appendix

Appendix A. AWS Development Network

1. AWS Marketplace : ICON Development Network
2. AWS CloudFormation
3. Interacting with Development Network

Appendix B. Development Resources

AWS Marketplace

AWS Marketplace - ICON Development Network

Why?

- Easy to install
- Scalable to 4 ~ 12 nodes
- Provides various monitoring tools

AWS Marketplace

1. Visit Marketplace [Link : Click]

aws marketplace

Hello, test.bc

Categories | Delivery Methods | Solutions | Migration Mapping Assistant | Your Saved List | Partners | Sell in AWS Marketplace | Amazon Web Services Home | Help

ICONLOOP

ICON Development Network

By: Icon Foundation Latest Version: 1.1

ICON Development Network will allow developers to easily run a private instance of the ICON Blockchain Network through AWS.

Linux/Unix (0)

Continue to Subscribe

Save to List

Typical Total Price

$0.192/hr

Total pricing per instance for services hosted on m5.xlarge in US East (N. Virginia). View Details

Overview | Pricing | Usage | Support | Reviews

Product Overview

ICON Development Network platform will provide an easy-to-use development-ready environment. This enables developers to bootstrap their own private ICON Network, adapted to their own convenience and need, in order to build, test or validate their project running on top of the ICON Protocol.
A private network built via AWS Cloud could be used to operate its own service as well in the context of a DApp and its production environment.

Version	1.1
By	Icon Foundation
Categories	Application Development
Operating System	Linux/Unix, Amazon Linux 2018.06.22
Delivery Methods	CloudFormation Template

Highlights

- Easy to install
- Scalable to 4 ~ 12 nodes
- Provides various monitoring tools

Pricing Information

Use this tool to estimate the software and infrastructure costs based on your configuration choices. Your usage and costs might be different from this estimate. They will be reflected on your monthly AWS billing reports.

Estimating your costs

Choose your region and fulfillment option to see the pricing details. Then,

AWS Marketplace

2. Go to Usage Information

aws marketplace

Hello, test.bc

Categories | Delivery Methods | Solutions | Migration Mapping Assistant | Your Saved List | Partners | Sell in AWS Marketplace | Amazon Web Services Home | Help

ICON Development Network

Continue to Subscribe

Overview | Pricing | Usage | Support | Reviews

Usage Information

Fulfillment Options

ICON DevNet Setup
CloudFormation Template

LoopChain Log Server, Monitoring Server, RadioStation Server, Peer Server Setup

View Template Components

View Usage Instructions

View CloudFormation Template

End User License Agreement

By subscribing to this product you agree to terms and conditions outlined in the product End User License Agreement (EULA)

Additional Resources

How it Works

CloudFormation Template

AWS CloudFormation templates are JSON or YAML formatted text files that simplify provisioning and management on AWS. The templates describe the service or application architecture you want to deploy and AWS CloudFormation uses those templates to provision and configure the required services (such as Amazon EC2 instances or Amazon RDS DB instances). The deployed application and associated resources is called a "stack".
Learn more

Support Information

ICON Development Network

Please allow 24 hours

https://www.icondev.io

AWS Infrastructure

AWS Support is a one-on-one support channel that is staffed 24x7x365 with experienced

Additional Resources

AWS Infrastructure Support

AWS Marketplace

3. Download CloudFormation Template

aws marketplace

Hello, test.bc

Categories | Delivery Methods | Solutions | Migration Mapping Assistant | Your Saved List | Partners | Sell in AWS Marketplace | Amazon Web Services Home | Help

ICON Development Network

Continue to Subscribe

Overview | Pricing | Usage | Support | Reviews

Usage Information

Fulfillment Options

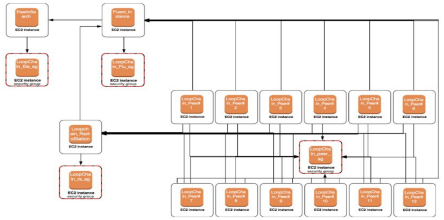
ICON DevNet Setup
CloudFormation Template

LoopChain Log Server, Monitoring Server, RadioStation Server, Peer Server Setup

View Template Components

View Usage Instructions

Close CloudFormation Template



Download CloudFormation Template
View Template in CloudFormation Designer

End User License Agreement

By subscribing to this product you agree to terms and conditions outlined in the product End User License Agreement (EULA)

Additional Resources

How it Works

CloudFormation Template

AWS CloudFormation templates are JSON or YAML formatted text files that simplify provisioning and management on AWS. The templates describe the service or application architecture you want to deploy and AWS CloudFormation uses those templates to provision and configure the required services (such as Amazon EC2 instances or Amazon RDS DB instances). The deployed application and associated resources is called a "stack".
Learn more

Dive into ICON - Appendix

Appendix A. AWS Development Network

1. AWS Marketplace : ICON Development Network
2. AWS CloudFormation
3. Interacting with Development Network

Appendix B. Development Resources

AWS CloudFormation

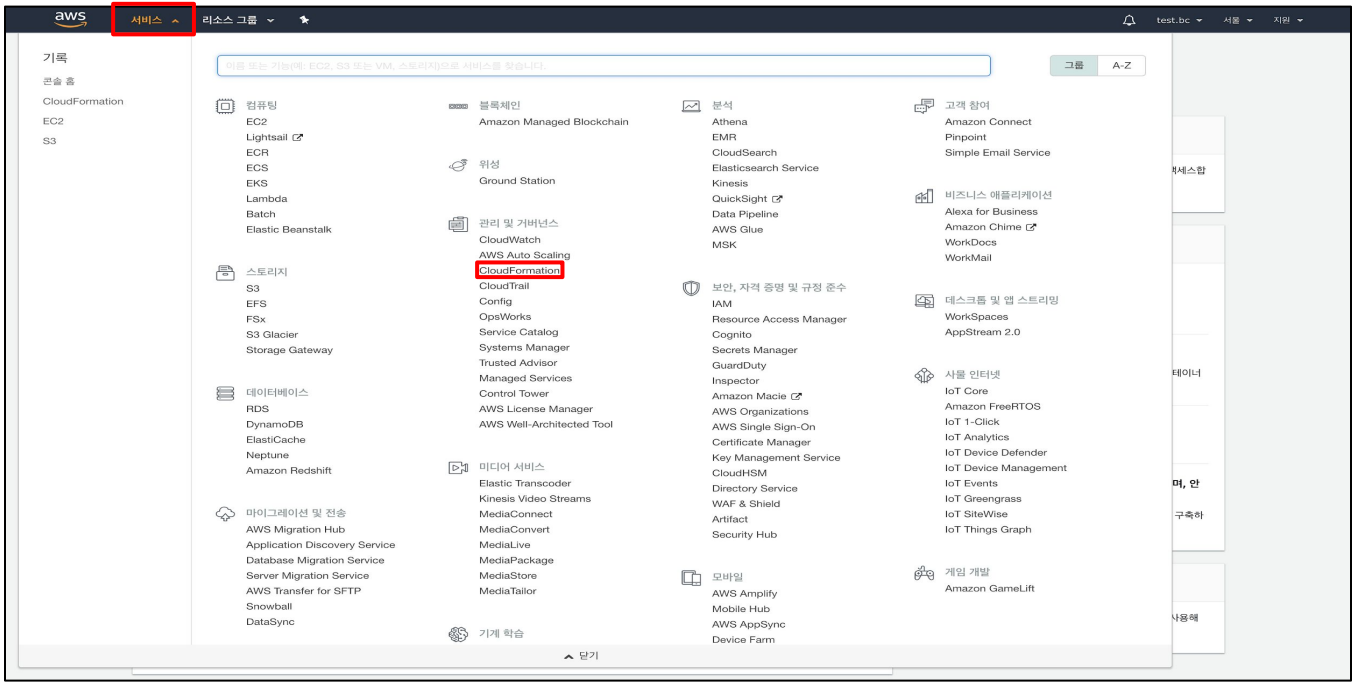
AWS CloudFormation - Create AWS Stack

Resource

- CloudFormation Template - ICON Development Network
- e.g) icon_dev_network.template

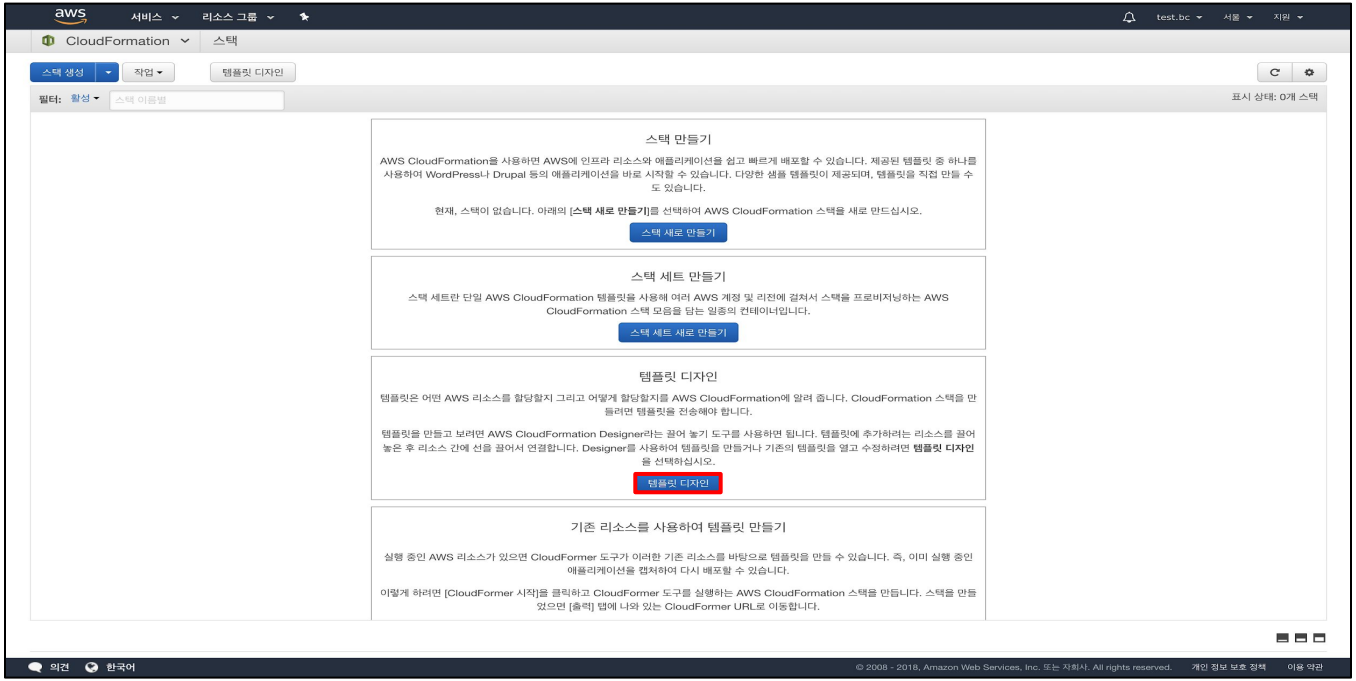
AWS CloudFormation

1. AWS Console ⇒ Service ⇒ CloudFormation



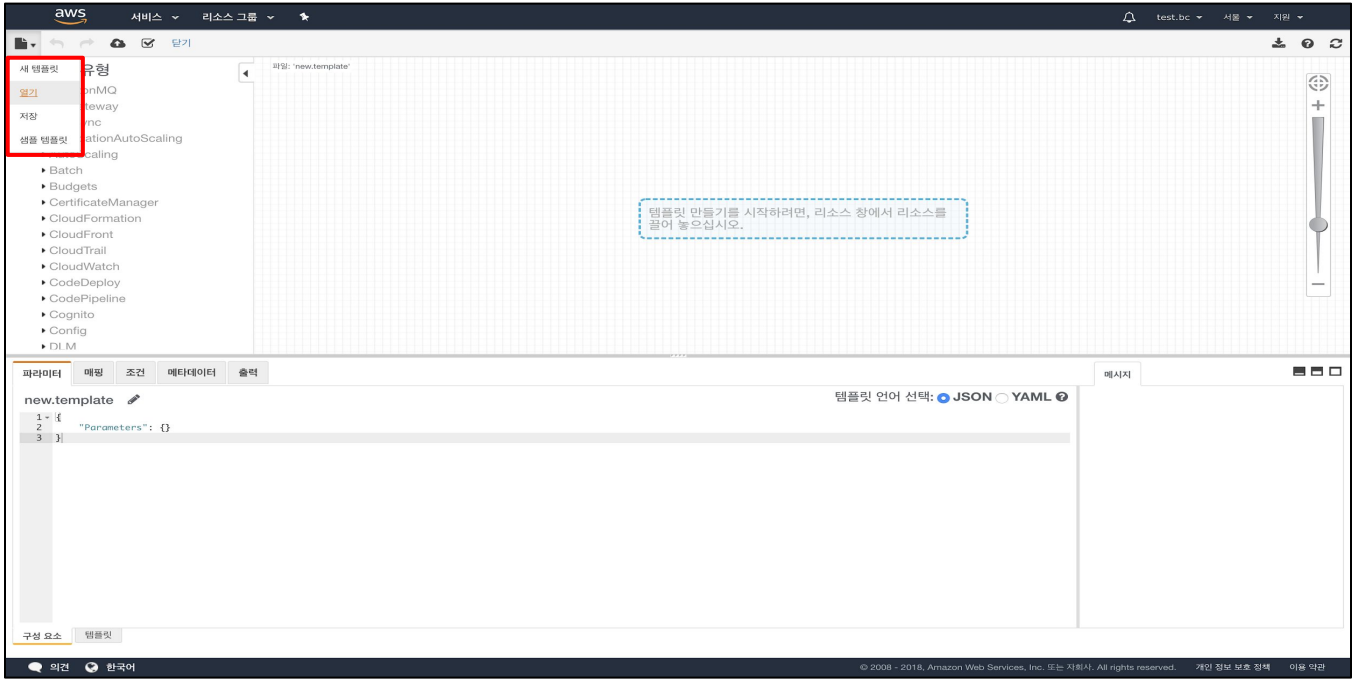
AWS CloudFormation

2. Click Template Design



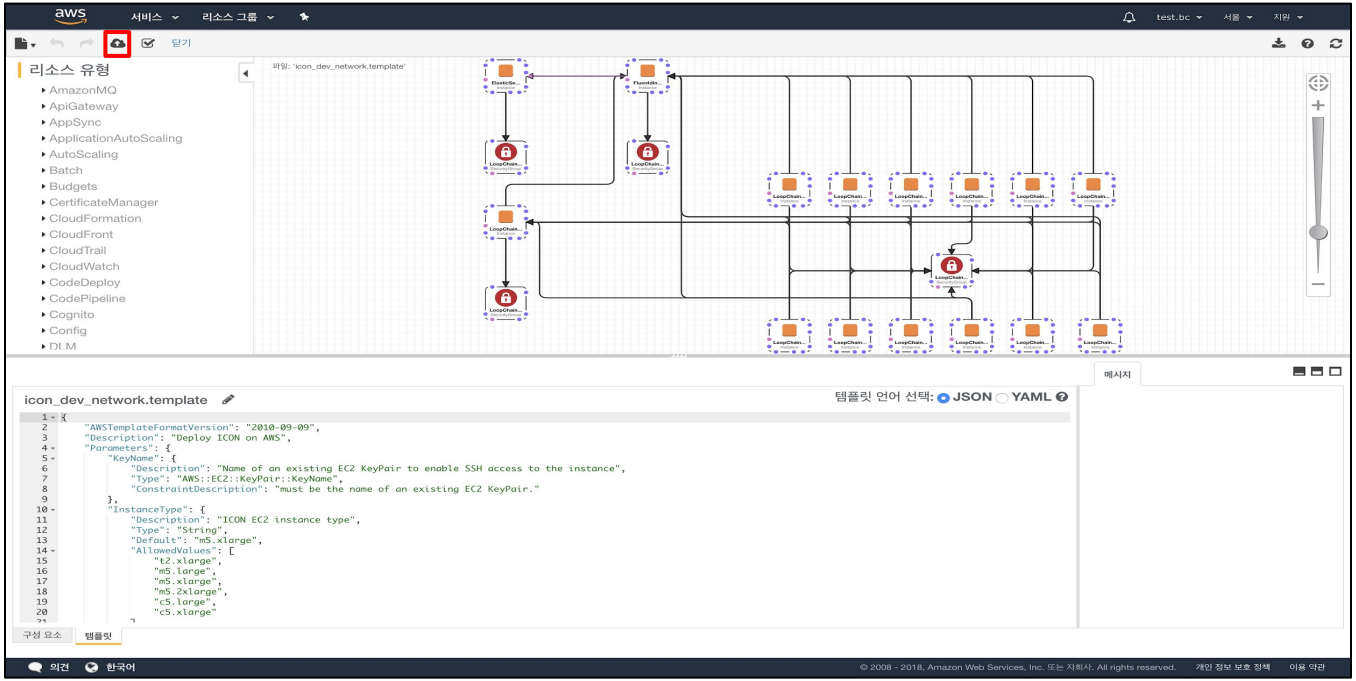
AWS CloudFormation

3. Open “icon_dev_network.template” on local storage



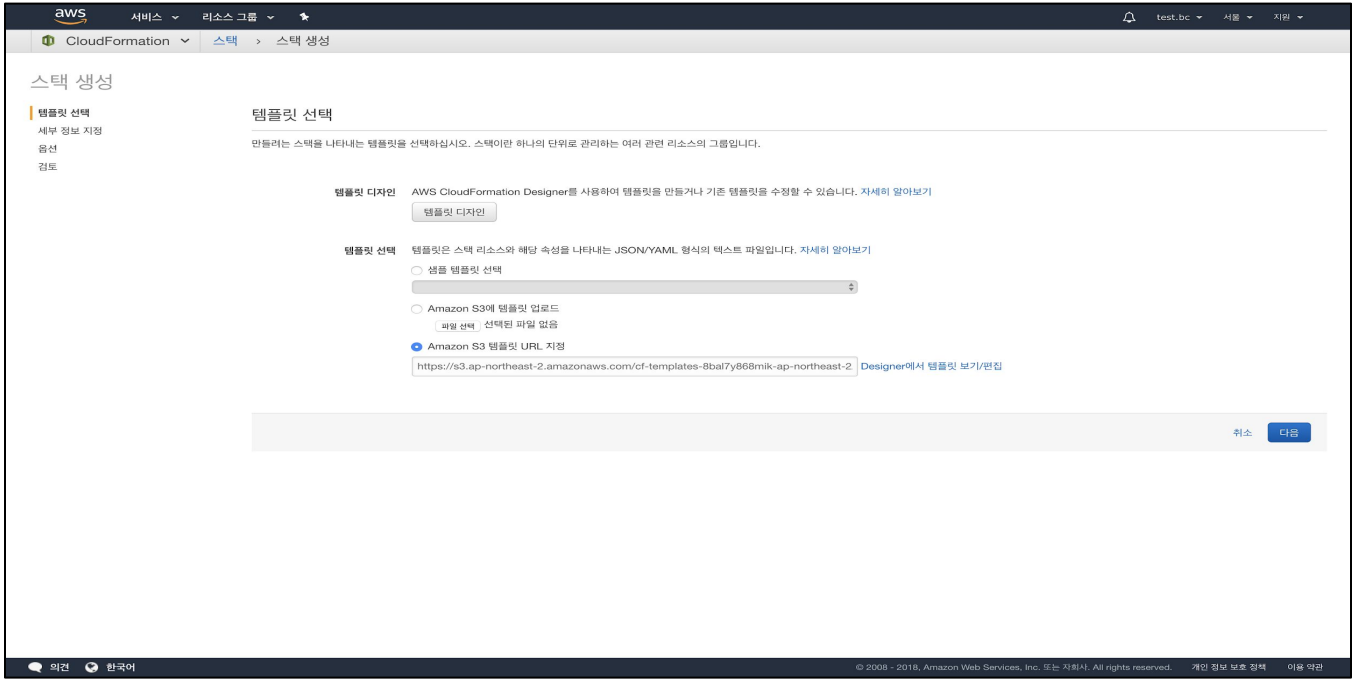
AWS CloudFormation

4. Create AWS Stack



AWS CloudFormation

5. Select CloudFormation Template ⇒ Next



AWS CloudFormation

6. Edit Configuration

템플릿 선택
세부 정보 지정
옵션
검토

세부 정보 지정

스택 이름과 파라미터 값을 지정하십시오. AWS CloudFormation 템플릿에 정의된 기본 파라미터 값을 사용하거나 변경할 수 있습니다. 자세히 알아보기

스택 이름

파라미터

AWS Instance and Network Settings

InstanceType	m5.xlarge	ICON EC2 instance type
NodeCount	4	ICON Node Cluster size; must be between 4 and 12
InstanceName	icon	EC2 Instance Name Prefix
VpcId	ID 또는 이름 태그 값으로 검색 VpcId of your existing Virtual Private Cloud (VPC)	
SubnetId	ID 또는 이름 태그 값으로 검색 Subnet should be a public subnet.	

Fee, Audit Parameters

Fee	false	Pay commission to execute the transaction by ICX or not
Audit	false	Prevent to deploy SCORE by anybody or not

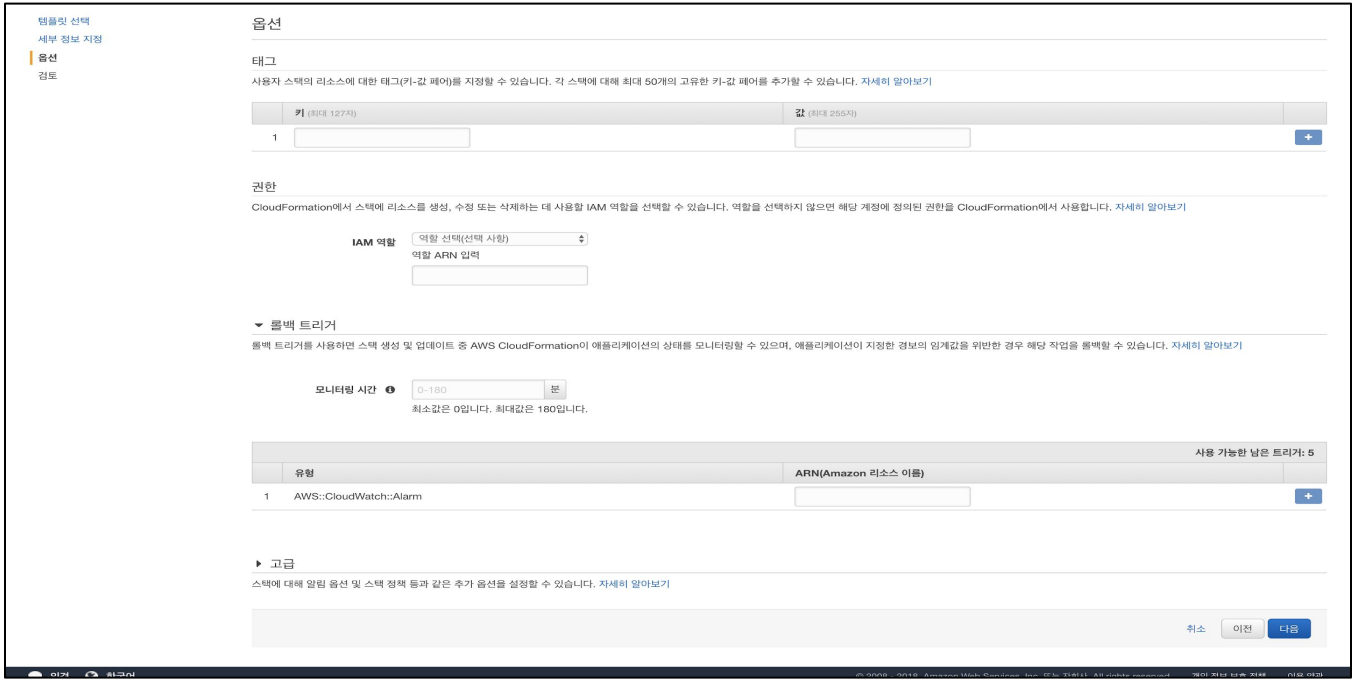
EC2 Parameters

KeyName	검색 Name of an existing EC2 KeyPair to enable SSH access to the instance	
SSHLocation		The IP address range that can be used to SSH to the EC2 instances

취소 이전 다음

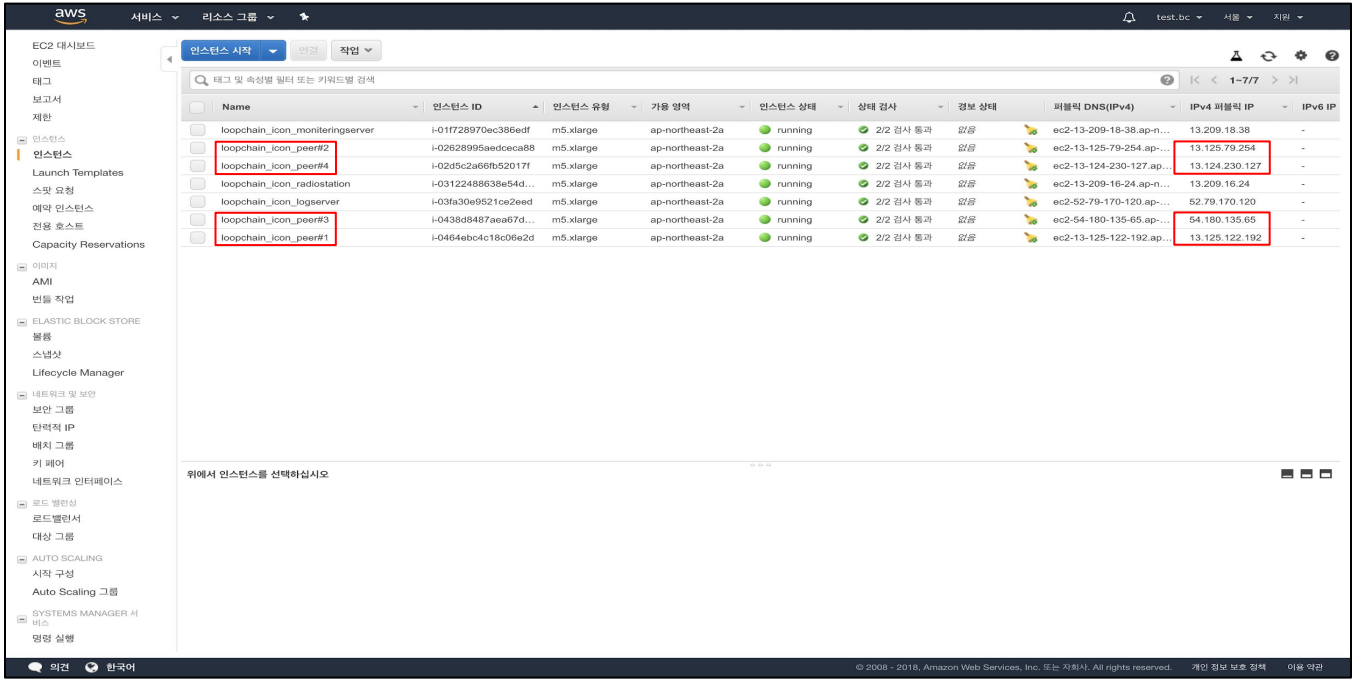
AWS CloudFormation

7. Finalize Options



AWS CloudFormation

8. Check Loaded Instances



Dive into ICON - Appendix

Appendix A. AWS Development Network

1. AWS Marketplace : ICON Development Network
2. AWS CloudFormation
3. Interacting with Development Network

Appendix B. Development Resources

Interacting with Development Network

How to use ICON Development Network

- Usage Instructions :
  - Access the application via a browser at http://<radiostation_dns>:9002/api/v1/peer/list.
  - To connect to the operating system, use SSH and the username ec2-user.

- JSON-RPC API v3 Document :
  https://github.com/icon-project/icon-rpc-server/blob/master/docs/icon-json-rpc-v3.md

- DApp Developer Guide : https://www.icondev.io

Interacting with Development Network

Send Query to Each Nodes - sendQuery.sh

```
#!/usr/bin/env zsh

echo "-----peer#1-----"
echo "Total Supply"
tbears totalsupply -u http://{peer#1 URL}:9000/api/v3
echo "----------------"
echo "Last Block"
tbears lastblock -u http://{peer#1 URL}:9000/api/v3


echo "-----peer#2-----"
echo "Total Supply"
tbears totalsupply -u http://{peer#2 URL}:9000/api/v3
echo "----------------"
echo "Last Block"
tbears lastblock -u http://{peer#2 URL}:9000/api/v3

...
```

Interacting with Development Network

Send Query to Each Nodes - sendQuery.sh

```
...

echo "-----peer#3-----"
echo "Total Supply"
tbears totalsupply -u http://{peer#3 URL}:9000/api/v3
echo "----------------"
echo "Last Block"
tbears lastblock -u http://{peer#3 URL}:9000/api/v3


echo "-----peer#4-----"
echo "Total Supply"
tbears totalsupply -u http://{peer#4 URL}:9000/api/v3
echo "----------------"
echo "Last Block"
tbears lastblock -u http://{peer#4 URL}:9000/api/v3
```

Interacting with Development Network

Query Results - **peer#1**

```
-----peer#1-----
Total Supply
Total supply of ICX in hex: 0x52c3ff441ba8feb88e00000
Total supply of ICX in decimal: 1600919800000000000000000000
------
Last Block
block info : {
    "jsonrpc": "2.0",
    "result": {
        ...
         "merkle_tree_root_hash": "72722dad5bdb0adff8f5fbb061de07a2404b522d463b708d1a62071262616fb3",
        ...
         "block_hash": "b3c99e161f8013b7dabc2ff8be56bacb3209853d139bb54f0a37384388a36505",
        ...
    },
    "id": 1
}
```

Interacting with Development Network

Query Results - **peer#2**

```
-----peer#2-----
Total Supply
Total supply of ICX in hex: 0x52c3ff441ba8feb88e00000
Total supply of ICX in decimal: 1600919800000000000000000000
------
Last Block
block info : {
    "jsonrpc": "2.0",
    "result": {
        ...
         "merkle_tree_root_hash": "72722dad5bdb0adff8f5fbb061de07a2404b522d463b708d1a62071262616fb3",
        ...
         "block_hash": "b3c99e161f8013b7dabc2ff8be56bacb3209853d139bb54f0a37384388a36505",
        ...
    },
    "id": 1
}
```

Interacting with Development Network

Query Results - **peer#3**

```
-----peer#3-----
Total Supply
Total supply of ICX in hex: 0x52c3ff441ba8feb88e00000
Total supply of ICX in decimal: 1600919800000000000000000000
------
Last Block
block info : {
    "jsonrpc": "2.0",
    "result": {
        ...
         "merkle_tree_root_hash": "72722dad5bdb0adff8f5fbb061de07a2404b522d463b708d1a62071262616fb3",
        ...
         "block_hash": "b3c99e161f8013b7dabc2ff8be56bacb3209853d139bb54f0a37384388a36505",
        ...
    },
    "id": 1
}
```

Interacting with Development Network

Query Results - **peer#4**

```
-----peer#4-----
Total Supply
Total supply of ICX in hex: 0x52c3ff441ba8feb88e00000
Total supply of ICX in decimal: 1600919800000000000000000000
------
Last Block
block info : {
    "jsonrpc": "2.0",
    "result": {
        ...
         "merkle_tree_root_hash": "72722dad5bdb0adff8f5fbb061de07a2404b522d463b708d1a62071262616fb3",
        ...
         "block_hash": "b3c99e161f8013b7dabc2ff8be56bacb3209853d139bb54f0a37384388a36505",
        ...
    },
    "id": 1
}
```

Summary

- AWS Marketplace

  https://aws.amazon.com/marketplace/pp/B07KBTZHZD?qid=1544602499452&sr=0-1&ref_=srh_res_product_title

- AWS CloudFormation
  - Create AWS Stack & Load Instances
  - Resource : CloudFormation Template

Appendix B. Development Resources

Dive into ICON - Appendix

Appendix A. AWS Development Network

Appendix B. Development Resources

Development Resources

- GitHub
- Developer Portal
- ICON official Documentation Project
- ICON Improvement Proposal

GitHub https://github.com/icon-project

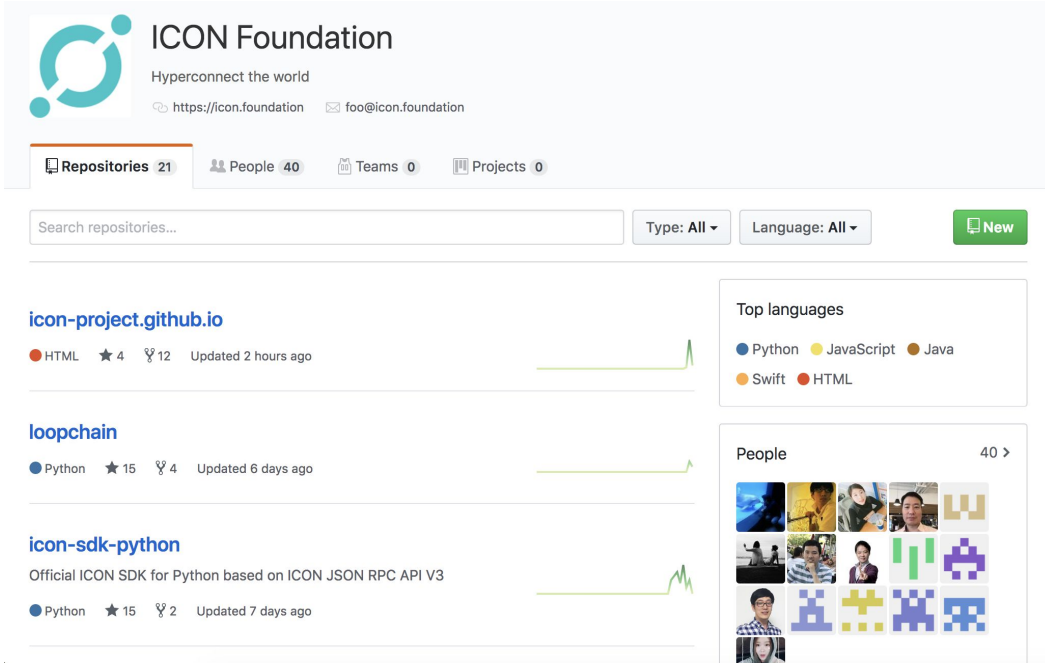
- loopchain
- icon-service
- icon-rpc-server

Node

- t-bears
- icon-sdk-python
- icon-sdk-java
- icon-sdk-js

Dev tools

- iconex_android
- iconex_ios
- iconex_chrome_extension



Developer Portal https://www.icondev.io

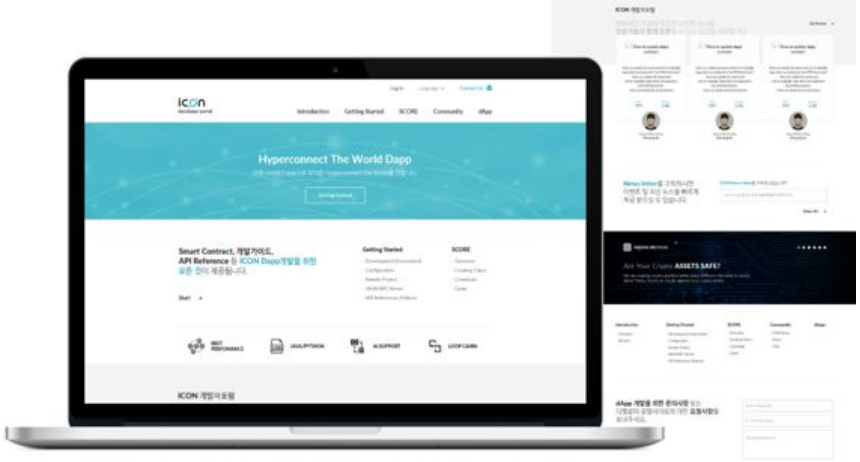
- Community portal for ICON DApp ecosystem

Getting Started
Tutorials for developers to get started

SCORE
Details on ICON's Smart Contract, SCORE

Community
Forum for Korean/English developers to discuss and communicate

DApp
Overview of ICON DApp Partners



ICON Improvement Proposal https://github.com/icon-project/IIPs

- IIP describes a standard for ICON platform.
- Anyone can prompt suggestions and discussions on new functions or improvement.
- Selected items will be implemented on ICON network.

- **For all other IIPs**, open a PR changing the state of your IIP to 'Final'. An editor will review your draft and ask if anyone objects to its being finalised. If the editor decides there is no rough consensus - for instance, because contributors point out significant issues with the IIP - they may close the PR and request that you fix the issues in the draft before trying again.

IIP Status Terms

- **Draft** - an IIP that is open for consideration.
- **Last Call** - an IIP that is calling for last review before finalizaing. IIPs that has been more than 2 weeks in Last Call without any technical changes or objections enters either Accepted or Final state.
- **Accepted** - an IIP that is planned for immediate adoption, i.e. expected to be included in the next release (for Core/Consensus layer IIPs only).
- **Final** - an IIP that has been adopted. For Core/Consensus layer IIPs, the implementation has been adopted in the mainnet.
- **Deferred** - an IIP that is not being considered for immediate adoption. May be reconsidered in the future.

IIPs

Number	Title	Author	Type	Status
1	IIP Purpose and Guidelines	Sojin Kim	Meta	Active
2	ICON Token Standard	Jaechang Namgoong	IRC	Final
3	ICON Non-Fungible Token Standard	Jaechang Namgoong	IRC	Draft
6	ICON Name Service Standard	Phyrex Tsai, Portal Network Team	IRC	Draft

Summary

- GitHub
- Developer Portal
- ICON official Documentation Project
- ICON Improvement Proposal