

# How to build DApps on Nebulas

Explore, Enjoy, Embrace

Roy Shang



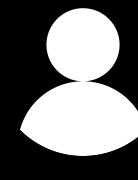
# Difference between DApps and Traditional Apps

## Nebulas

Manage Wallets  
Create new Wallets  
Unlock a Wallet  
View Wallet Info

Send Transactions  
Transfer Tokens  
Deploy Smart Contracts  
Execute Methods in Smart Contracts

Call Methods  
Call Methods in Smart Contracts



{ api }

## Traditional Platform

Manage Accounts  
Create new Accounts  
Login an Account  
View Account Profile

Submit Form  
Pay bills  
Deploy a new Application  
Stored Procedure in MySQL

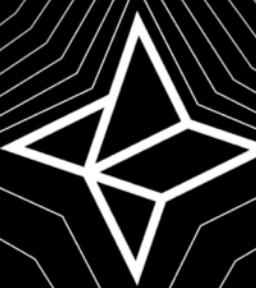
Invoke Read-Only API  
Fetch Data in the Application

# CONTENT

01 Wallet

02 Smart Contracts

03 DApps



**NEBULAS**

THINKING IN BLOCKCHAIN

01  
Wallet



# Create Wallet

<https://github.com/nebulasio/web-wallet>

A screenshot of the NEBULAS web wallet creation interface. At the top, there is a navigation bar with the NEBULAS logo, language selection (Local Nodes ▾ English ▾), and several buttons: Create New Wallet (underlined), Send NAS, Send Offline, View Wallet Info, Check TX Status, and Contract. Below the navigation bar, there is a password input field with placeholder text "Enter a password: (Not less than 9 characters)". A note "Do NOT forget to save this!" is displayed above the password field. A large black button labeled "Create New Wallet" is centered below the input field. At the bottom of the interface, there is a note: "This password encrypts your private key. This does not act as a seed to generate your keys. You will need this password + your private key to unlock your wallet."

NEBULAS

Local Nodes ▾ English ▾

Create New Wallet    Send NAS    Send Offline    View Wallet Info    Check TX Status    Contract

Enter a password: (Not less than 9 characters)

Do NOT forget to save this!

**Create New Wallet**

This password encrypts your private key.  
This does not act as a seed to generate your keys.  
You will need this password + your private key to unlock your wallet.

A wallet = A private key = keystore file + passphrase -> An address



# Prepare Tokens

Testnet: <https://testnet.nebulas.io/claim/>

A screenshot of the Nebulas Testnet token claiming interface. The page has a white background with a light gray header bar. At the top left is the Nebulas logo. Below it, the word "Testnet" is displayed. The main form area contains two input fields: "Email:" with placeholder "Your Email" and "Wallet:" with placeholder "Nebulas Wallet Address". Below these is a large black button with the text "Claim Token (1 NAS per email per day)". Further down, there is another input field for "Wallet:" with placeholder "Nebulas Wallet Address" and a second black button labeled "Check Balance". At the bottom of the page is a footer bar containing the Nebulas logo, navigation links for Home, Technology, Community, Team, Resources, and Blog, and a copyright notice: "Copyright © 2017 Nebulas.io, 814 Mission Street, San Francisco".

Mainnet: Exchanges

Huobi.pro



## Send Transaction – Unlock Wallet

The screenshot shows the NEBULAS wallet interface with the following elements:

- Top Bar:** NEBULAS logo, Network selection (1. Testnet), Language selection (English).
- Navigation:** Create New Wallet, Send NAS (underlined), Send Offline, View Wallet Info, Check TX Status, Contract.
- Form Fields:** Select Your Wallet File (input field), SELECT WALLET FILE... (button labeled 2.).
- Buttons:** Unlock (large black button).

1. Select Network: Mainnet, **Testnet** or LocalNet
2. Import **keystore file** & Unlock it with **passphrase**

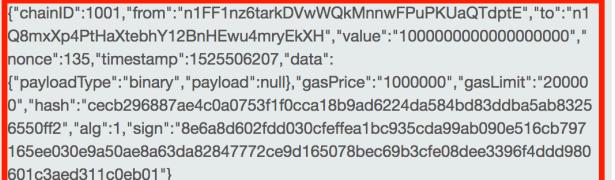
# Send Transaction – Arguments

From Address	Balance
n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE	 2. 6.966096946991 ≈ 7 NAS
To Address	Value / Amount to Send
1. n1Q8mxXp4PtHaXtebhY12BnHEwu4mryEkXH	 3. 1 ≈ 1 NAS
Gas Limit	Gas Price ( 1 NAS = 1EWei = $10^{18}$ Wei )
200000	1000000 ≈ 1 MWei
Nonce	
135	
<b>Generate Transaction</b>	

1. Fill the **To Address**
2. Check **Balance**
3. Fill the **Value**

# Send Transaction – Submit

**Raw Transaction**

1. 

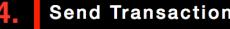
**Signed Transaction**

2. 

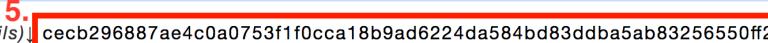
**Signed Transaction QR**

3. 

**Send Transaction**

4. 

**txhash : (Click to view transaction details)**

5. 

**receipt :**

```
{"hash": "cecb296887ae4c0a0753f1f0cca18b9ad6224da584bd83ddba5ab83256550ff2", "chainId": 1001, "from": "n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE", "to": "n1Q8mxXp4PtHaXtebhY12BnHEwu4mryEkXH", "value": "10000000000000000000000000000000", "nonce": "135", "timestamp": "1525506207", "type": "binary", "data": null, "gas_price": "100000", "gas_limit": "200000", "contract_address": "", "status": "2", "gas_used": ""}
```

1. **Transaction Arguments**
2. **Serialized Transaction ( base64 )**
3. **QR Code of Serialized Transaction**
4. Click **Send Transaction**, wait for a while, the transaction **receipt** will be returned identified by **txhash**
5. Click **txhash** to check the transaction **status**

# Send Transaction – Check Status

## Check TX Status

Trading hash can query transaction information, including pending and packaged transactions. You need to refresh the package status change of the query transaction several times when the transaction is packaged and validated.

cecb296887ae4c0a0753f1f0cca18b9ad6224da584bd83ddba5ab83256550ff2

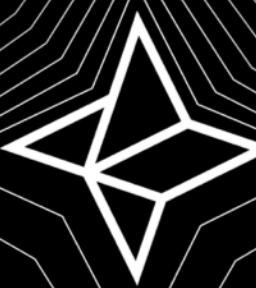
1. **Check TX Status**

### Transaction Details

TX Hash	cecb296887ae4c0a0753f1f0cca18b9ad6224da584bd83ddba5ab83256550ff2
Contract address	
TxReceipt Status	2. success
From Address	n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE
To Address	n1Q8mxXp4PtHaXtebhY12BnHEwu4mryEkXH

1. Refresh the status
2. In general, the status will become success after 15 seconds

If you are in the pending state for a long time  
Check the balance of the sending account  
Make sure it's enough to pay the gas fee



**NEBULAS**

THINKING IN BLOCKCHAIN

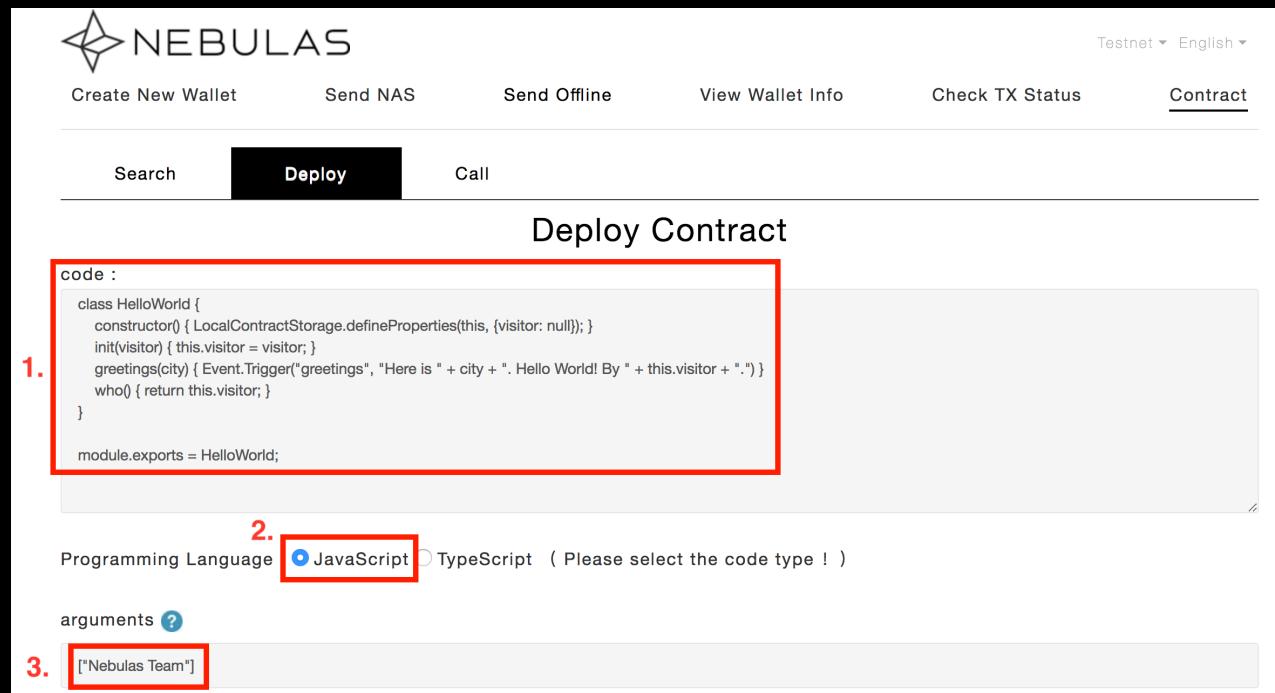
02

## Smart Contracts

## Hello World – Deploy - Write Smart Contract

```
class HelloWorld {  
    constructor() {  
        LocalContractStorage.defineProperties(this, {  
            visitor: null  
        });  
    }  
  
    init(visitor) {  
        this.visitor = visitor;  
    }  
  
    greetings(city) {  
        Event.Trigger("greetings", "Here is " + city + ". Hello World! By " + this.visitor + ".")  
    }  
  
    who() {  
        return this.visitor;  
    }  
}  
  
module.exports = HelloWorld;
```

# Hello World – Deploy - Write Smart Contract



1. When the contract is **deployed**, it is executed by default:
  - Contract constructor function: **constructor**
  - Contract initialization function: **init**
2. Select **JavaScript**
3. Fill in initialization parameters, array form [A, B, ...]

# Hello World – Deploy - Unlock Wallet

Select Your Wallet File:

```
n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE
```

Your wallet is encrypted. Good! Please enter the password.

.....

1. **Unlock**

From Address	To Address
<pre>n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE</pre>	<pre>n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE</pre>

Balance

5.966096926991 NAS

Gas Limit

200000 ≈ 200 k

Value / Amount to Send

0

Gas Price ( 1 NAS = 1EWei =  $10^{18}$  Wei )

1000000 ≈ 200 kWei

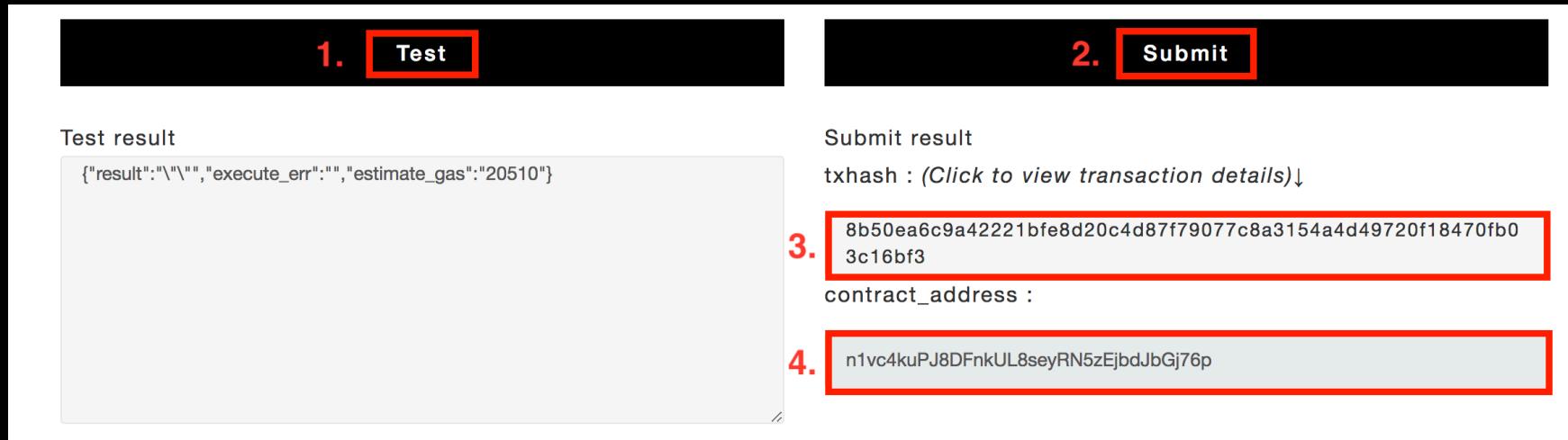
**Test**

**Submit**

**Deploy Contracts**  
= Send Transactions

**From Address = To Address**

# Hello World – Deploy - Test & Submit



1. **Test Button:** Simulate the execution of **constructor** and **init** functions
2. **Submit Button:** Submit the contract to Nebulas platform, return **txhash** and **contract address**
3. **Txhash:** Click it the check the transaction status
4. **Contract Address:** Used as To Address in executing contracts

# Hello World – Deploy - Check

**Check TX Status**

Trading hash can query transaction information, including pending and packaged transactions. You need to refresh the package status change of the query transaction several times when the transaction is packaged and validated.

8b50ea6c9a42221bfe8d20c4d87f79077c8a3154a4d49720f18470fb03c16bf3

1. **Check TX Status**

**Transaction Details**

TX Hash	8b50ea6c9a42221bfe8d20c4d87f79077c8a3154a4d49720f18470fb03c16bf3
Contract address	n1vc4kuPJ8DFnkUL8seyRN5zEjbdJbGj76p
TxReceipt Status	2. <b>success</b>
From Address	n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE
To Address	n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE

8b50ea6c9a42221bfe8d20c4d87f79077c8a3154a4d49720f18470fb03c16bf3

1. Refresh the status
2. In general, the status will become success after 15 seconds

If you are in the **pending** state for a long time

Check the balance of the sending account  
Make sure it's enough to pay the **gas fee**

If you are in the **fail** state

Check the transaction using **Test button** again

# Hello World – Execute - Method



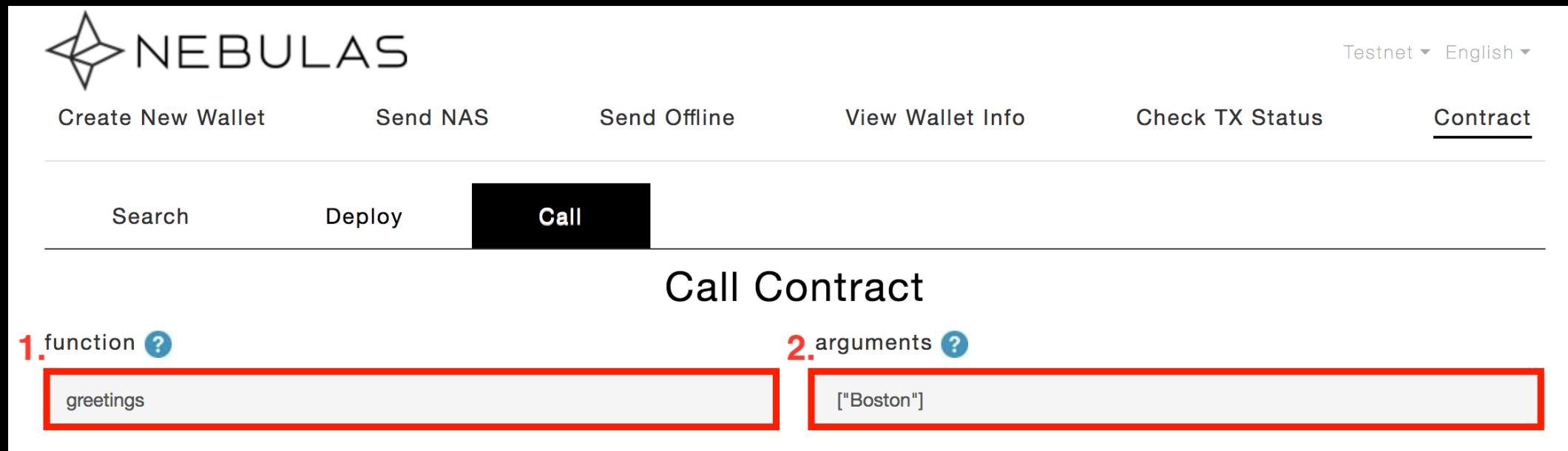
The screenshot shows the NEBULAS wallet interface on the Testnet. The top navigation bar includes options like 'Create New Wallet', 'Send NAS', 'Send Offline', 'View Wallet Info', 'Check TX Status', and 'Contract'. The 'Contract' tab is selected. Below it, there are three buttons: 'Search' (highlighted in red), 'Deploy', and 'Call'. A sub-menu titled 'Search Contract' is open, showing a single result: '8b50ea6c9a42221bfe8d20c4d87f79077c8a3154a4d49720f18470fb03c16bf3'. A red box highlights this entry. Below the search results is a black 'Search' button. The main content area displays the source code of the HelloWorld contract:

```
class HelloWorld {
    constructor() {
        LocalContractStorage.defineProperties(this, {
            visitor: null
        });
    }
    init(visitor) {
        this.visitor = visitor;
    }
    greetings(city) {
        Event.Trigger("greetings", "Here is " + city + ". Hello World! By " + this.visitor + ".");
    }
    who() {
        return this.visitor;
    }
}

module.exports = HelloWorld;
```

1. Search contracts by deployment Txhash
2. Prepare to execute the **greetings** function in the contract  
- Record an event on chain

# Hello World – Execute – Method Arguments



The screenshot shows the NEBULAS wallet interface with the 'Contract' tab selected. Below it, the 'Call' button is highlighted. The 'Call Contract' section contains two input fields. The first field, labeled '1. function ?' with 'greetings' inside, is highlighted with a red box. The second field, labeled '2. arguments ?' with '["Boston"]' inside, is also highlighted with a red box.

1. Fill in the function **name**
2. Fill in the function **parameters**

# Hello World – Execute - Contract Address

Select Your Wallet File:

n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE

Your wallet is encrypted. Good! Please enter the password.

.....

1. **Unlock**

From Address: n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE

To Address: n1vc4kuPJ8DFnkUL8seyRN5zEjbdJbGj76p

Balance: 5.966096906481 NAS

Value / Amount to Send: 0

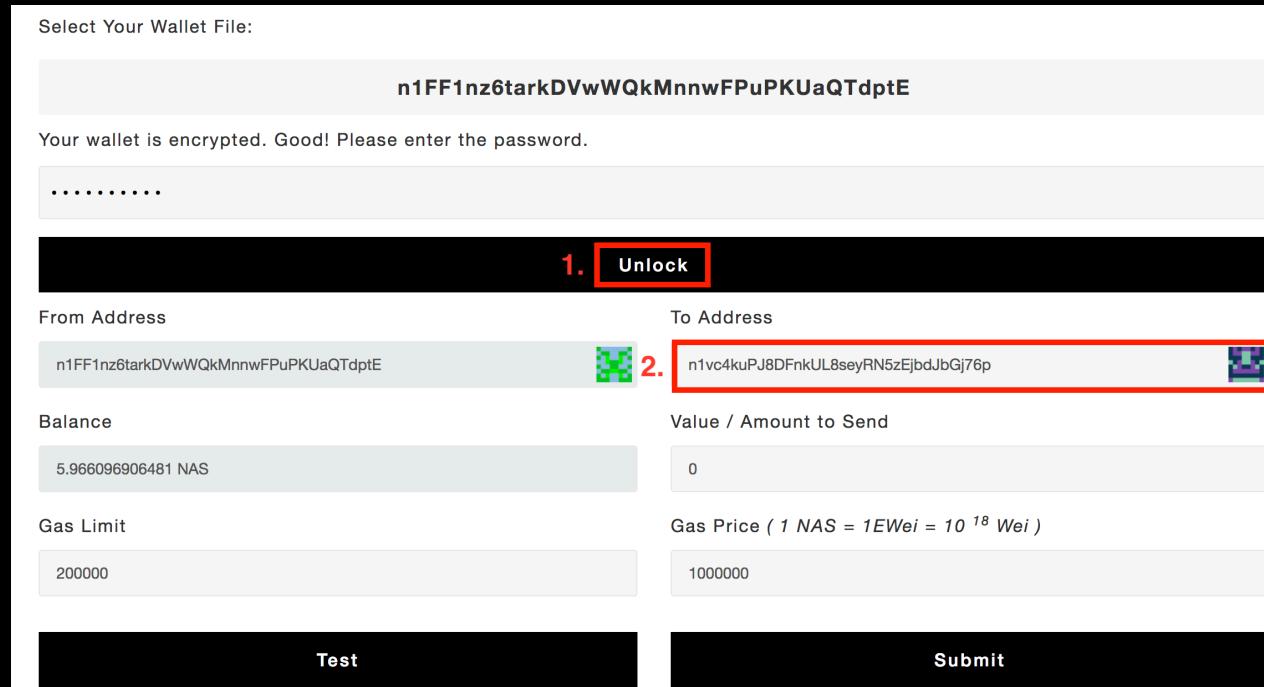
Gas Limit: 200000

Gas Price ( 1 NAS = 1EWei =  $10^{18}$  Wei )

1000000

Test

Submit

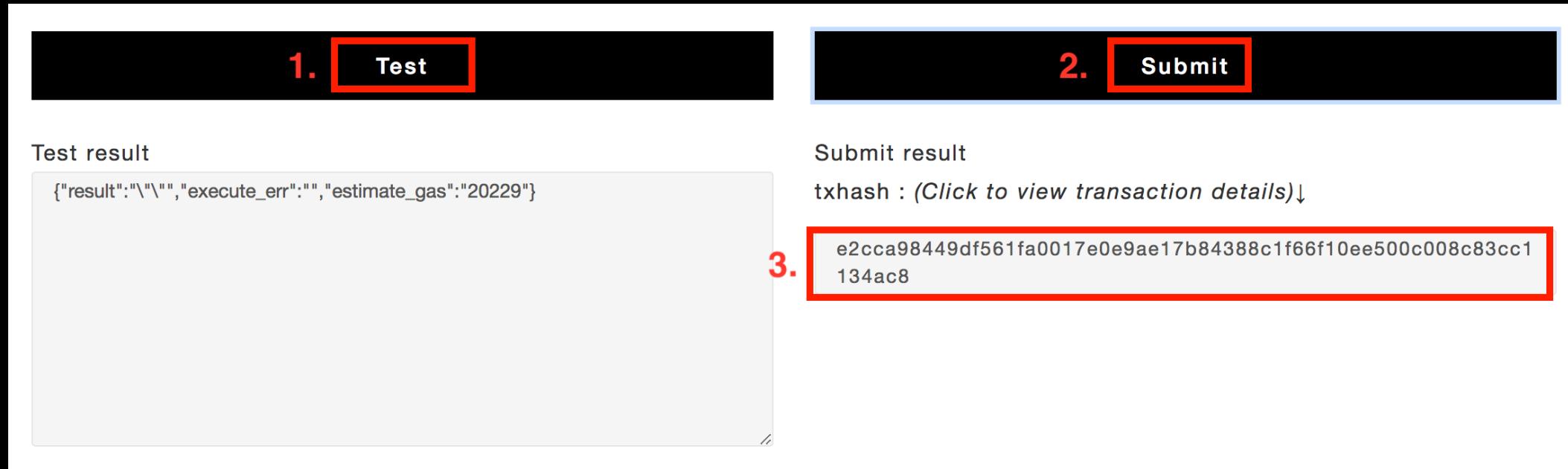


1. **Unlock wallet**

2. **Fill in contract address**

n1vc4kuPJ8DFnkUL8seyRN5zEjbdJbGj76p

## Hello World – Execute - Test & Submit



The screenshot shows a user interface for executing a Hello World smart contract. It consists of two main sections: 'Test result' on the left and 'Submit result' on the right.

**Test result:** A black button labeled '1. Test' is highlighted with a red border. Below it, the text 'Test result' is displayed, followed by a JSON object: {"result": "\\"", "execute\_err": "", "estimate\_gas": "20229"}.

**Submit result:** A black button labeled '2. Submit' is highlighted with a red border. Below it, the text 'Submit result' is displayed, followed by 'txhash : (Click to view transaction details)↓'. A red box highlights the txhash value: e2cca98449df561fa0017e0e9ae17b84388c1f66f10ee500c008c83cc1134ac8.

1. **Test Button:** Simulate the execution of greetings function
2. **Submit Button:** Submit the execution to Nebulas platform, return **txhash**
3. **Txhash:** Click to check the transaction status

# Hello World – Execute - Check

**Check TX Status**

Trading hash can query transaction information, including pending and packaged transactions. You need to refresh the package status change of the query transaction several times when the transaction is packaged and validated.

e2cca98449df561fa0017e0e9ae17b84388c1f66f10ee500c008c83cc1134ac8

1. **Check TX Status**

**Transaction Details**

TX Hash	e2cca98449df561fa0017e0e9ae17b84388c1f66f10ee500c008c83cc1134ac8
Contract address	
TxReceipt Status	2. success
From Address	n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE
To Address	n1vc4kuPJ8DFnkUL8seyRN5zEjbdJbGj76p

e2cca98449df561fa0017e0e9ae17b84388c1f66f10ee500c008c83cc1134ac8

1. Refresh the status
2. In general, the status will become success after 15 seconds

If you are in the pending state for a long time

Check the balance of the sending account  
Make sure it's enough to pay the gas fee

If you are in the fail state

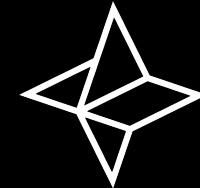
Check the transaction using Test button again

## Hello World – Execute - Event

```
> curl -i -H 'Content-Type: application/json' -X POST  
https://testnet.nebulas.io/v1/user/getEventsByHash  
-d '{"hash":"e2cc ... 4ac8"}'
```

←  
**Invoke API**

```
{  
  "result":{  
    "events": [  
      {  
        "topic": "chain.contract.greetings",  
        "data": "\\"Here is Boston. Hello World! By Nebulas Team.\\""  
      }  
      ,  
      {  
        "topic": "chain.transactionResult",  
        "data": "{\"hash\":\"e2cc...4ac8\"},\"status\":1,\"gas_used\":\"20229\"},\"error\":\"\\""  
      }  
    ]  
  }  
}
```



NEBULAS  
THINKING IN BLOCKCHAIN

# Hello World – Call - Methods



The screenshot shows the NEBULAS wallet interface with the 'Contract' tab selected. A red box highlights the search bar containing the deployment Txhash: 8b50ea6c9a42221bfe8d20c4d87f79077c8a3154a4d49720f18470fb03c16bf3. Below the search bar is a large code editor window displaying the Hello World contract source code. A red box highlights the `who()` method definition.

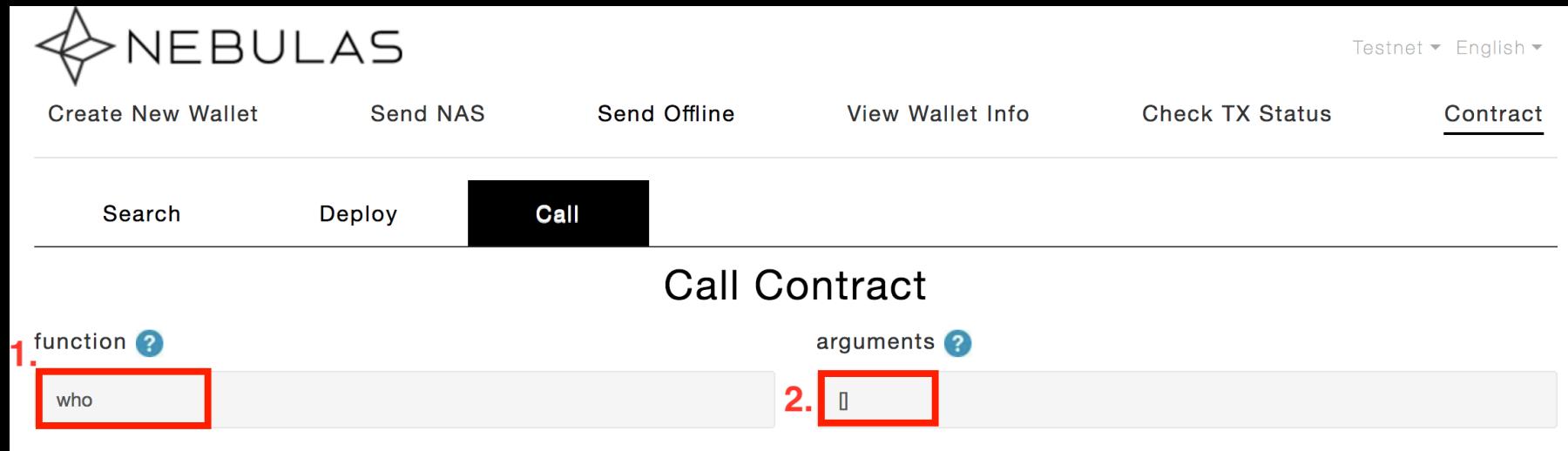
```
1. Search Contract
8b50ea6c9a42221bfe8d20c4d87f79077c8a3154a4d49720f18470fb03c16bf3
Search

class HelloWorld {
  constructor() {
    LocalContractStorage.defineProperties(this, {
      visitor: null
    });
  }
  init(visitor) {
    this.visitor = visitor;
  }
  greetings(city) {
    Event.Trigger("greetings", "Here is " + city + ". Hello World! By " + this.visitor + ".");
  }
  2. who() {
    return this.visitor;
  }
}

module.exports = HelloWorld;
```

1. Search contracts by deployment Txhash
2. Prepare to call the `who` function in the contract
  - Return `visitor` assigned in `init` function

# Hello World – Call – Method Arguments



The screenshot shows the NEBULAS wallet interface with the 'Contract' tab selected. The 'Call' button is highlighted. The 'Call Contract' section contains two input fields: 'function' with the value 'who' and 'arguments' with an empty value.

1. function ?  
2. arguments ?

1. Fill in function **name**
2. Fill in function **parameters**

# Hello World – Call - Contract Address

Select Your Wallet File:

```
n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE
```

Your wallet is encrypted. Good! Please enter the password.

.....

1. **Unlock**

From Address  
n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE

To Address  
n1vc4kuPJ8DFnkUL8seyRN5zEjbdJbGj76p

Balance  
5.966096886252 NAS

Value / Amount to Send  
0

Gas Limit  
200000

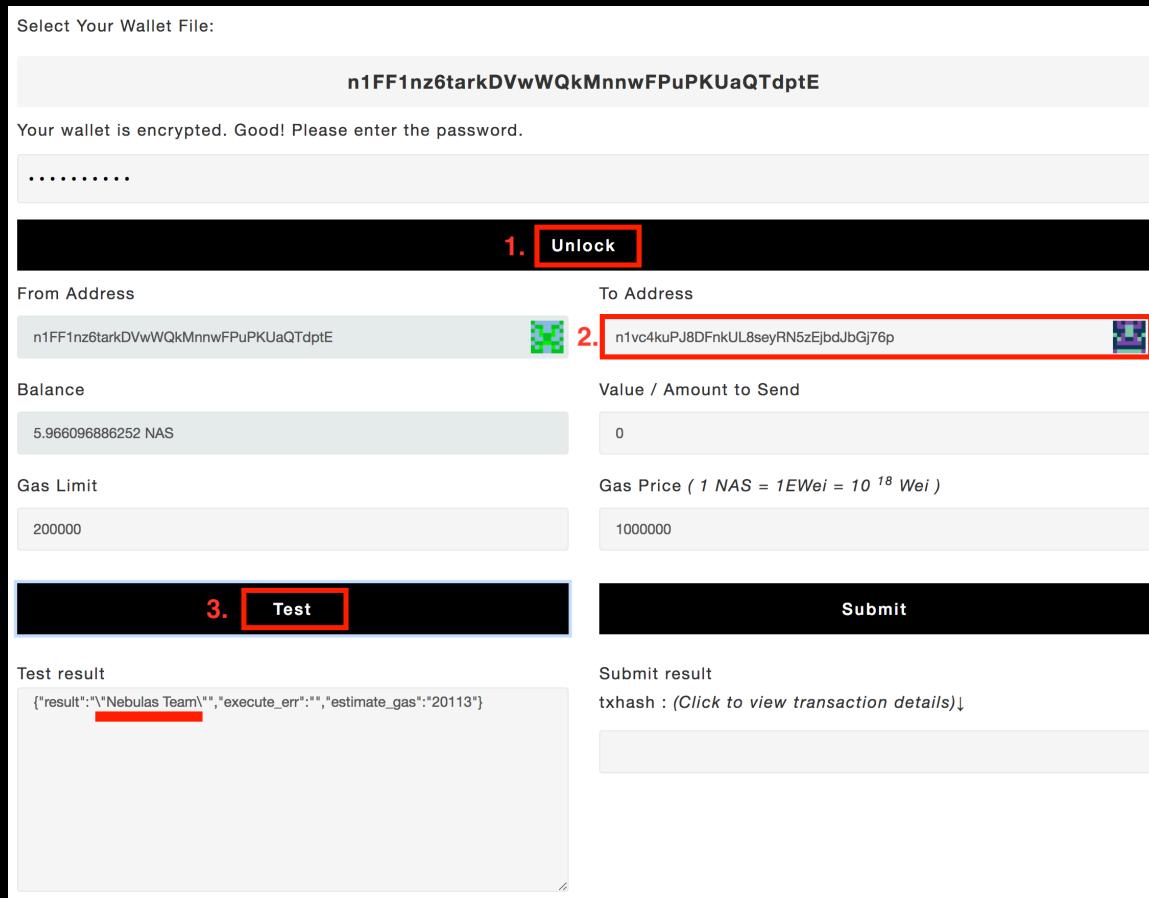
Gas Price ( 1 NAS = 1EWei =  $10^{18}$  Wei )  
1000000

3. **Test**

Submit

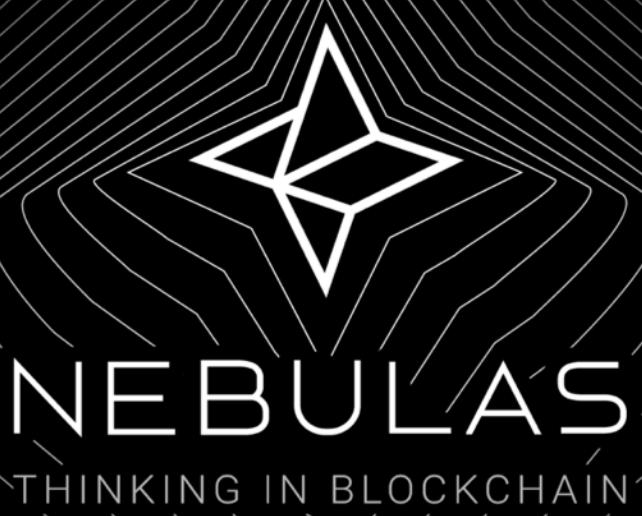
Test result  
{"result":"\\Nebulas Team\\", "execute\_err": "", "estimate\_gas": "20113"}

Submit result  
txhash : (Click to view transaction details)↓

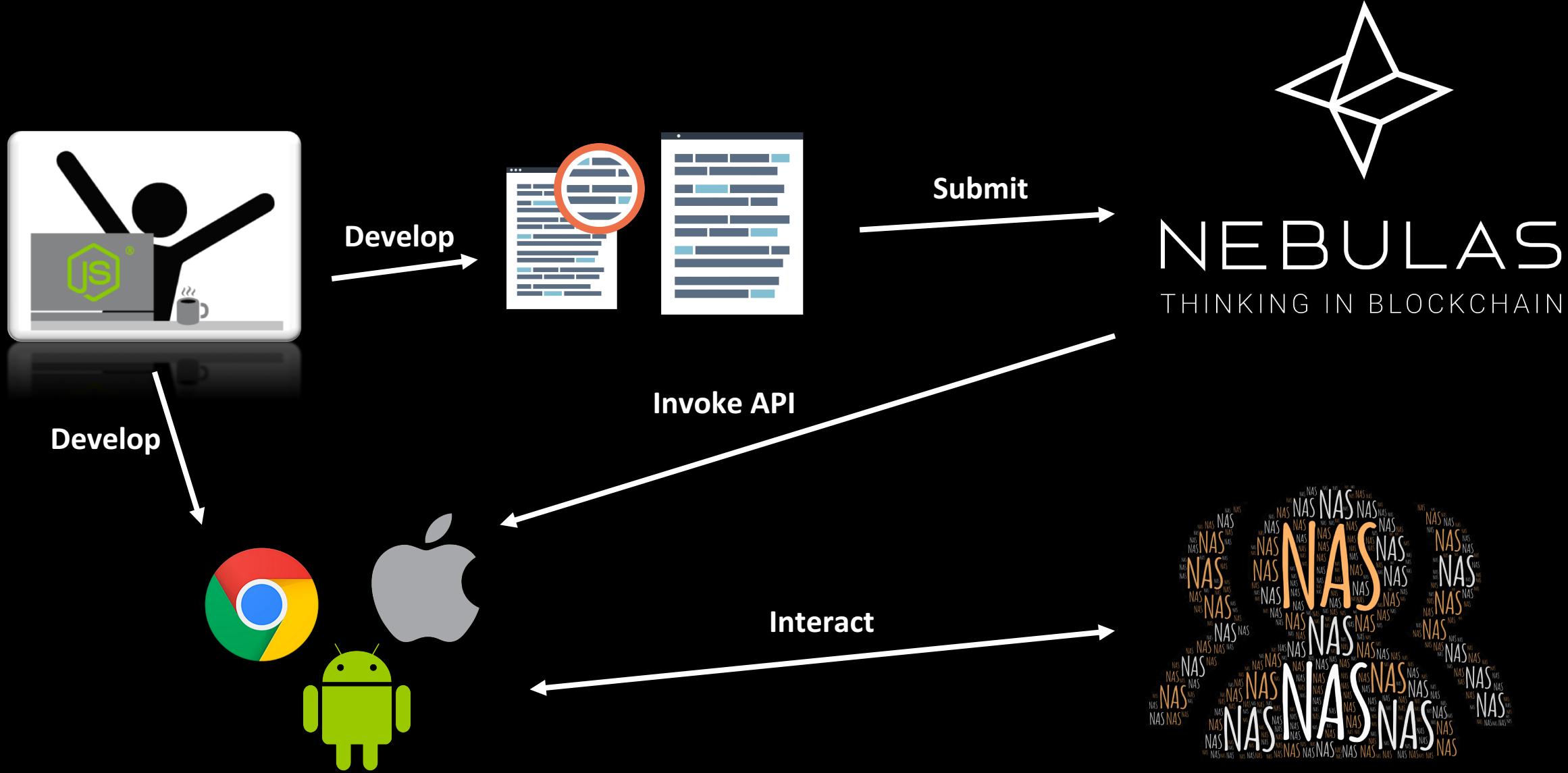


1. **Unlock wallet to get From Address**
2. **Fill contract address**
3. **Click Test button to get return value**

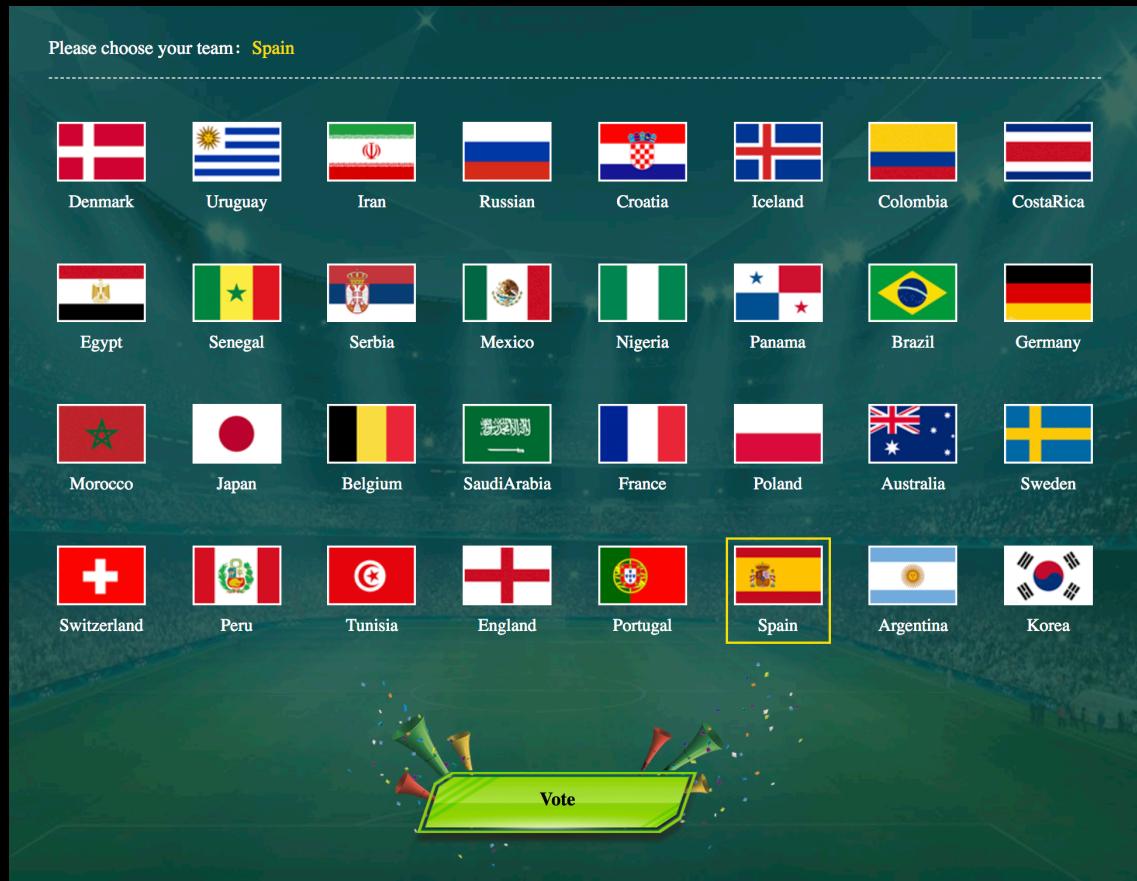
n1vc4kuPJ8DFnkUL8seyRN5zEjbdJbGj76p



03  
DApps



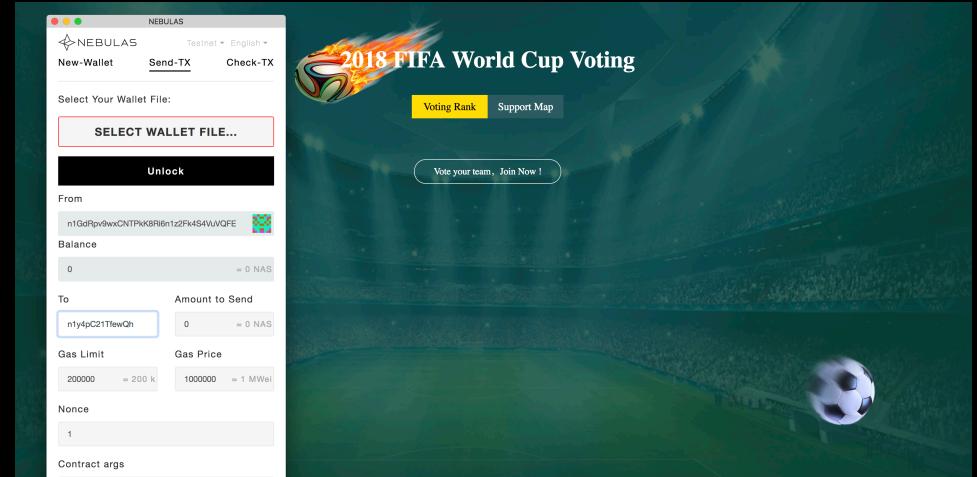
# Frontend



# User Wallets

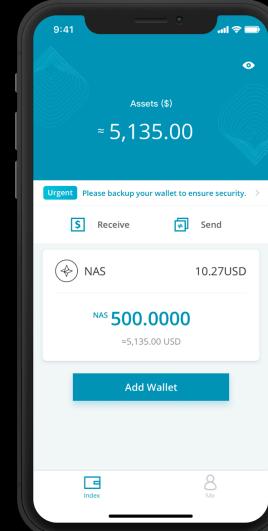
## Chrome Extension – Like MetaMask in Ethereum

- <https://github.com/ChengOrangeJu/WebExtensionWallet>
- Support Mainnet, Testnet and Localnet
- Platform: PC



## NebPay – Like ApplePay with Nebulas Mobile Wallet

- Wallet: [https://nano.nebulas.io/index\\_en.html](https://nano.nebulas.io/index_en.html)
- WebApp: <https://github.com/nebulasio/nebPay>
- IOS: <https://github.com/nebulasio/neb.iOS>
- Android: <https://github.com/nebulasio/neb.android>
- Support Mainnet
- Platform: PC & Mobile



<https://github.com/nebulasio/wiki/blob/master/tutorials.md>

All you need to build a Dapp can be found in this page.



 [nebulas.io](https://nebulas.io)

 [t.me/nebulasio](https://t.me/nebulasio)

 [nebulasio.herokuapp.com](https://nebulasio.herokuapp.com)

 [twitter.com/nebulasio](https://twitter.com/nebulasio)

 [/r/nebulas](https://www.reddit.com/r/nebulas)

 Channel: Nebulas

 [medium.com/nebulasio](https://medium.com/@nebulasio)

 [contact@nebulas.io](mailto:contact@nebulas.io)