HITACHI
Inspire the Next

Testing

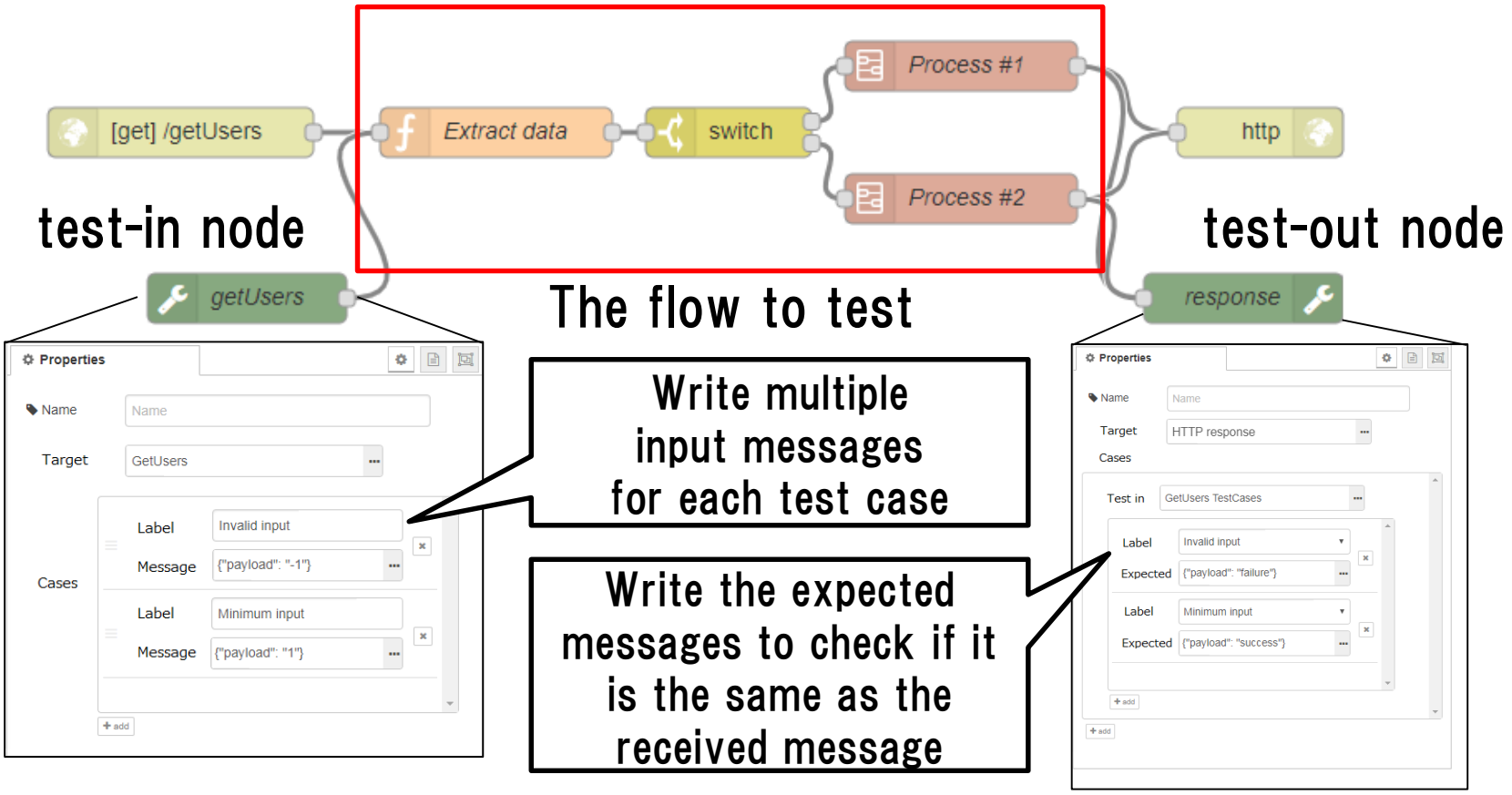
Kazuhito Yokoi

Framework for flow testing

HITACHI
Inspire the Next

Framework to create test cases of flows and execute them on flow editor

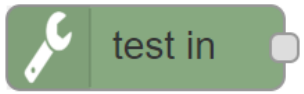
- The flow to test is wrapped by test-in and test-out node (like pair of http-in and http-response nodes)
- Testing flows from CLI is also supported



Test-in and test-out nodes

HITACHI
Inspire the Next

- **Test-in node**



The node to send messages instead of the input node

- **Test-out node**

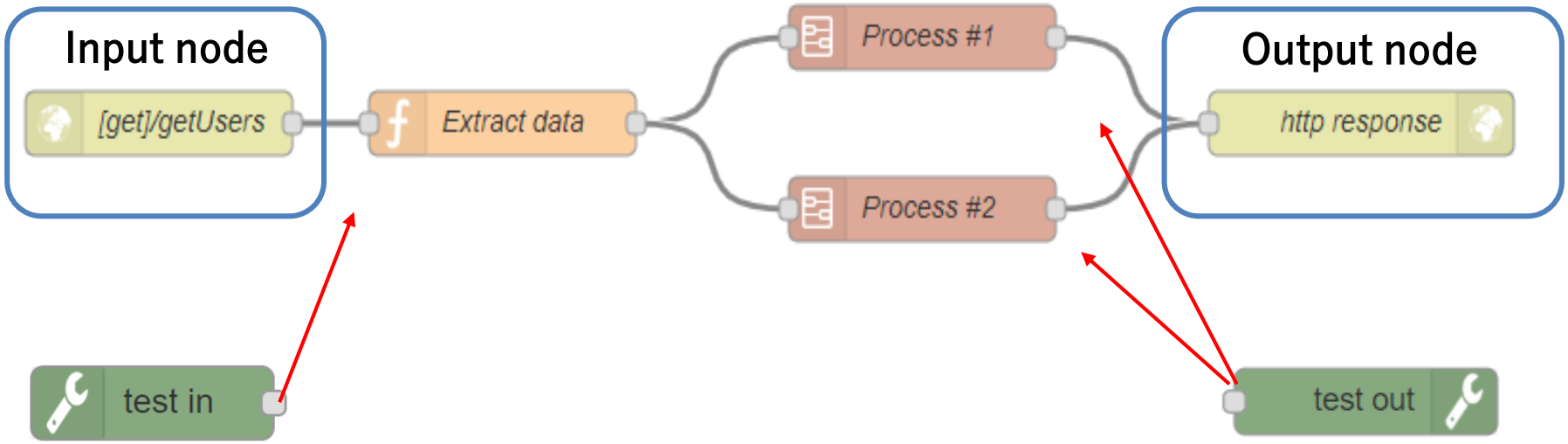
The node to receive message instead of output node and check that it is the expected message

How to use test-in and test-out nodes

HITACHI
Inspire the Next

- Test-in node:
  A user connects it to the input port of the first node in the flow to test.

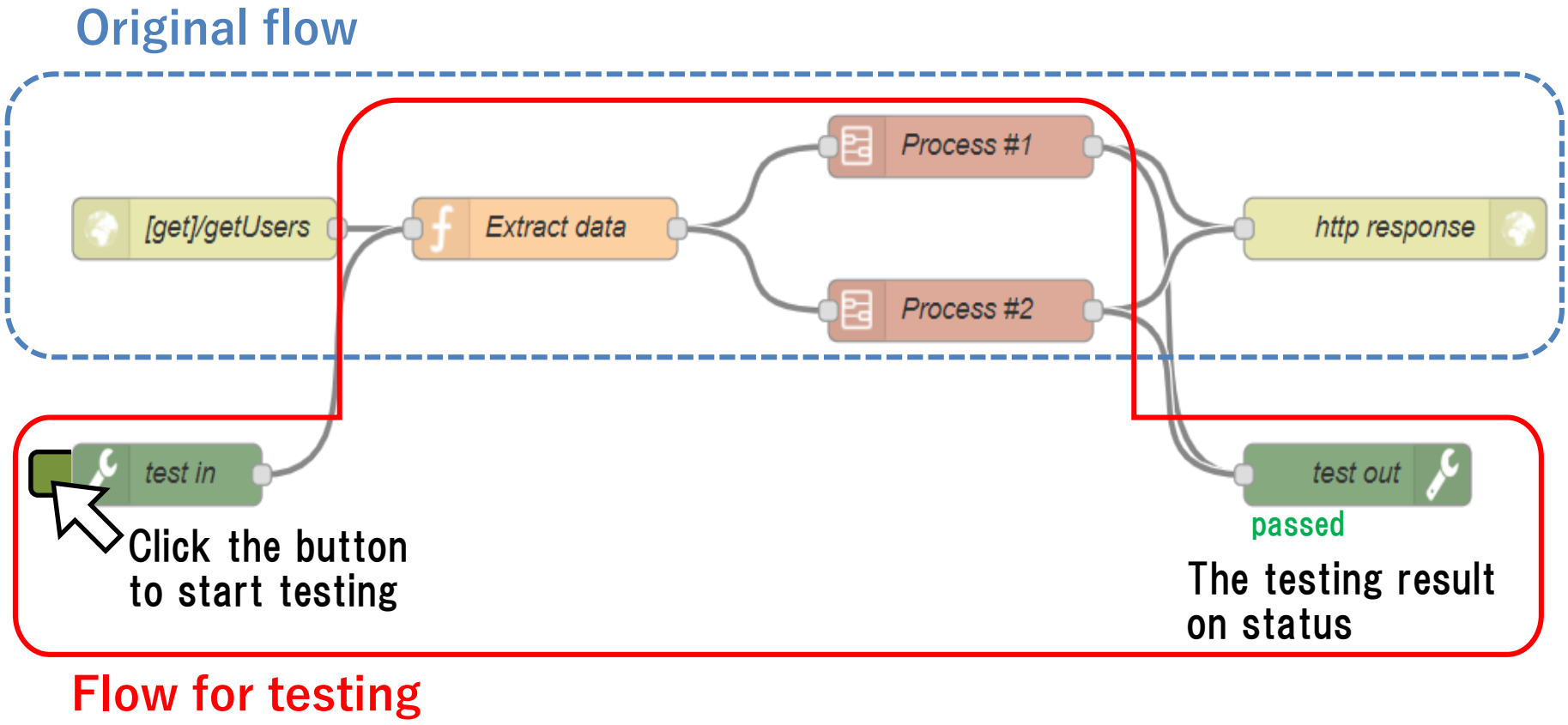
- Test-out node:
  A user connects it to the output port of the last node in the flow to test.



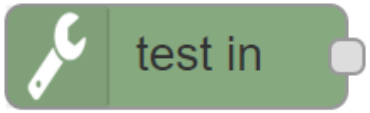
Execution of the flow testing

HITACHI
Inspire the Next

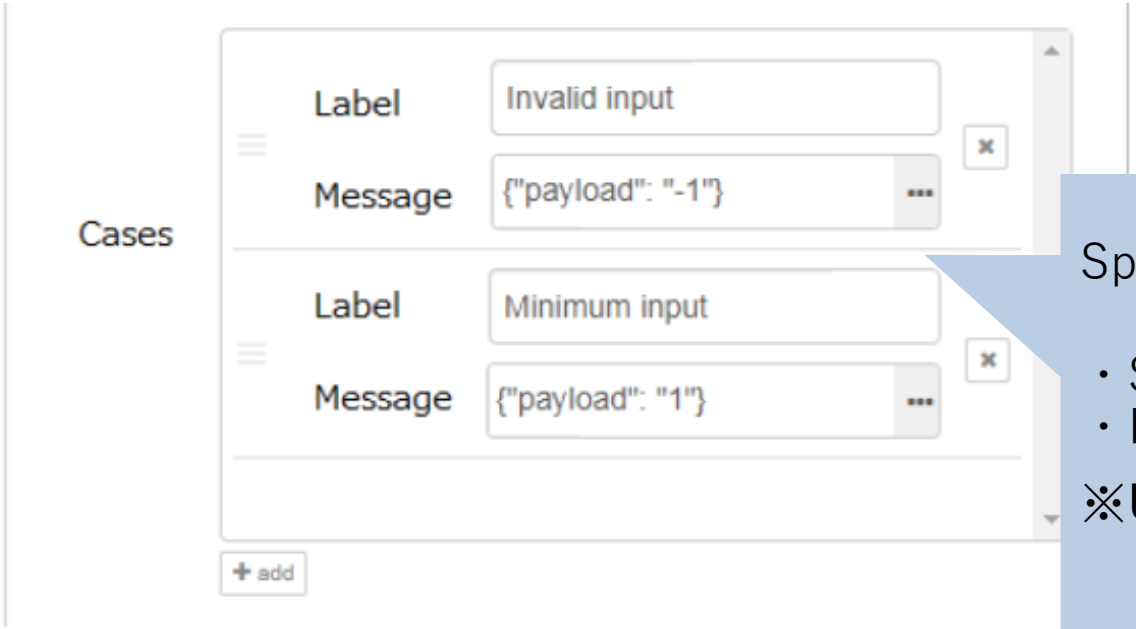
- To start flow testing, a user clicks the button of test-in node
- After execution of the flow testing,
  test-out has the result of testing in the status



Node property UI in Test-in node



The node which sends the messages for testing instead of the input node

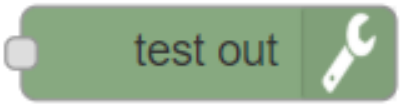


Specify test scenarios

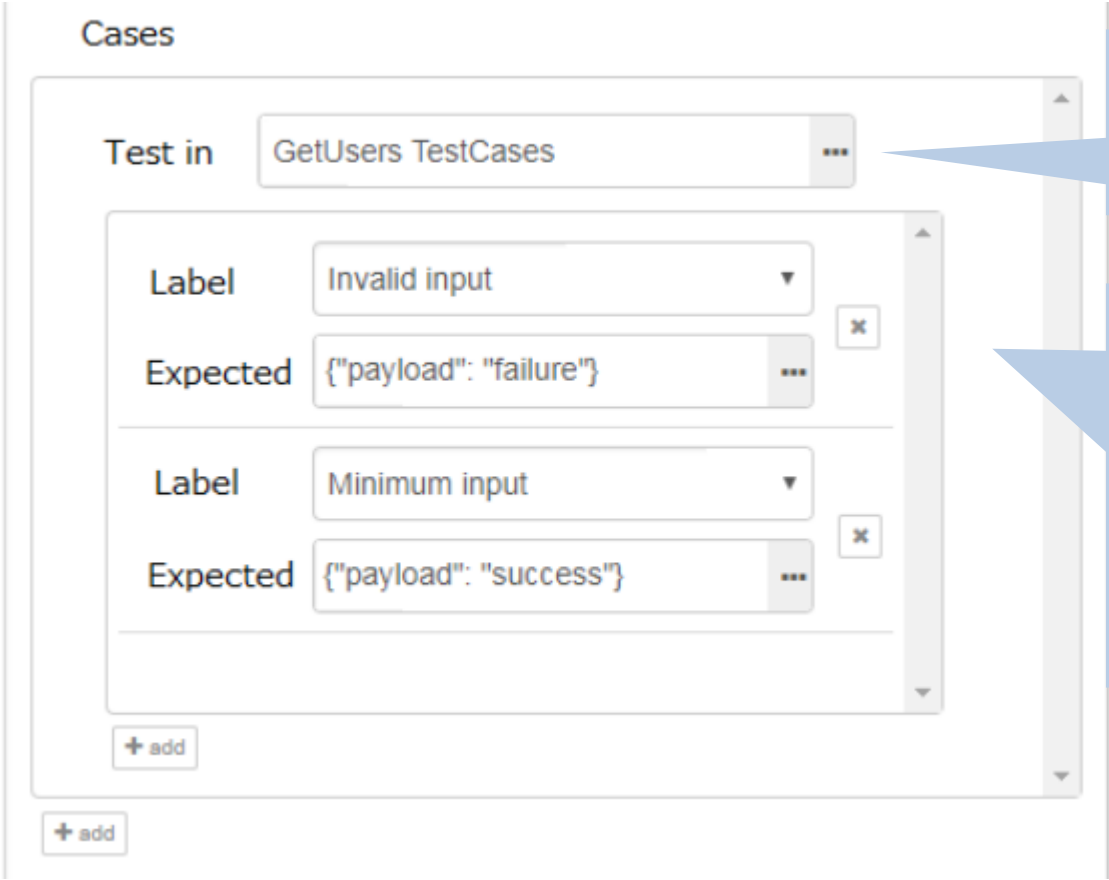
- Scenario name
- Input message

※**Users can register multiple scenarios in the UI**

Node property UI in test-out node



The node which receives message instead of an output node and checks the received message is the expected value



Specify test-in node which sends messages to this test-out node

Input the expected message for each scenario

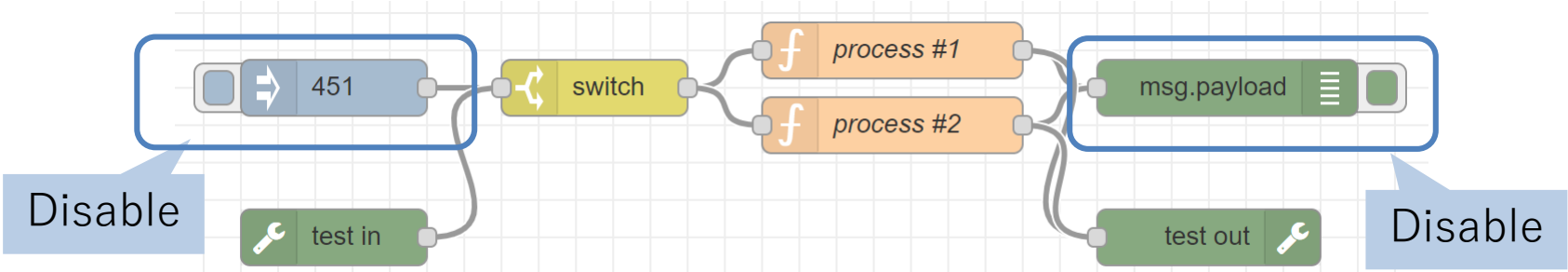
(If the received message equals the expected message, the testing process will be a success)

How to handle the first and the last nodes in the flow

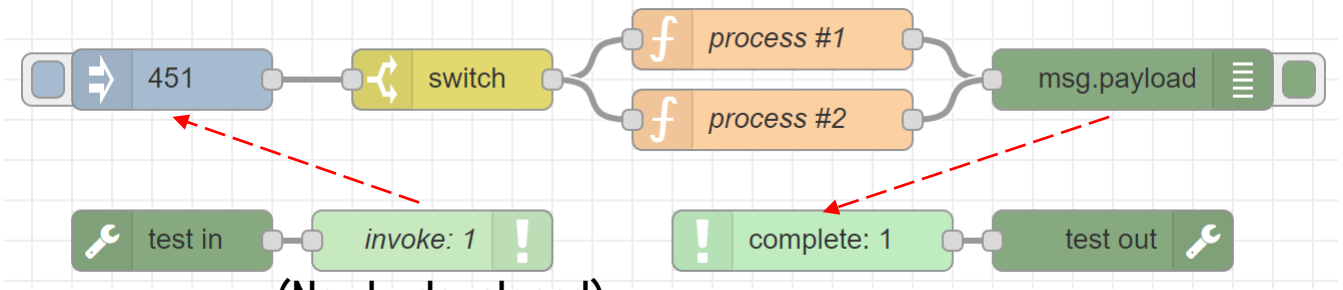
HITACHI
Inspire the Next

#	Suggestion
1	Flow testing supports both methods

- Method 1: Making the first and the last nodes disabled
(It will be suitable for internal logic)



- Method 2: The invoke node sends the test message to the first node.
The complete node receives the test message from the last node.
(This method can test entire flow including the first and last nodes)



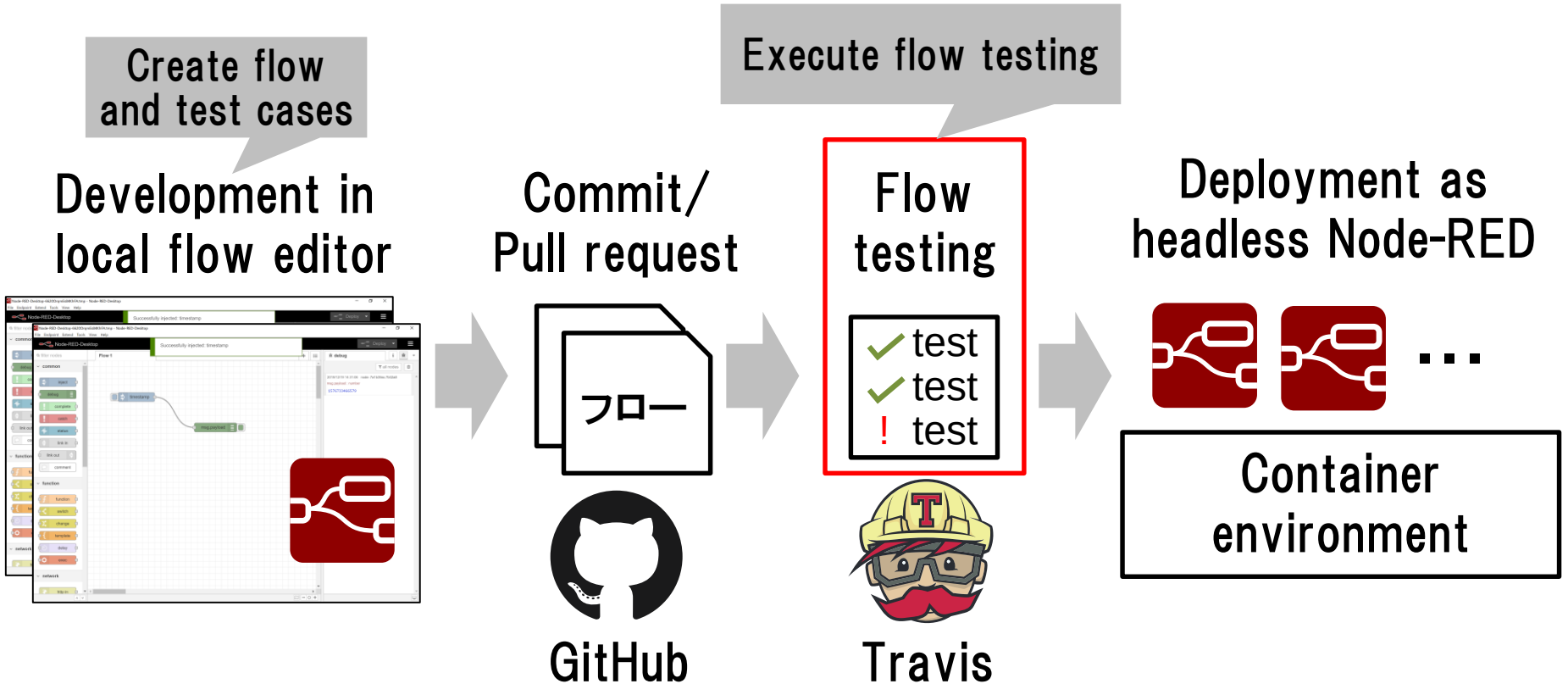
(Newly developed)

Execution of flow testing from command line

HITACHI
Inspire the Next

Flow testing from CLI will be required in the deployment pipeline.

- To check the flow in each commit
- To test the flow before deployment to the production environment



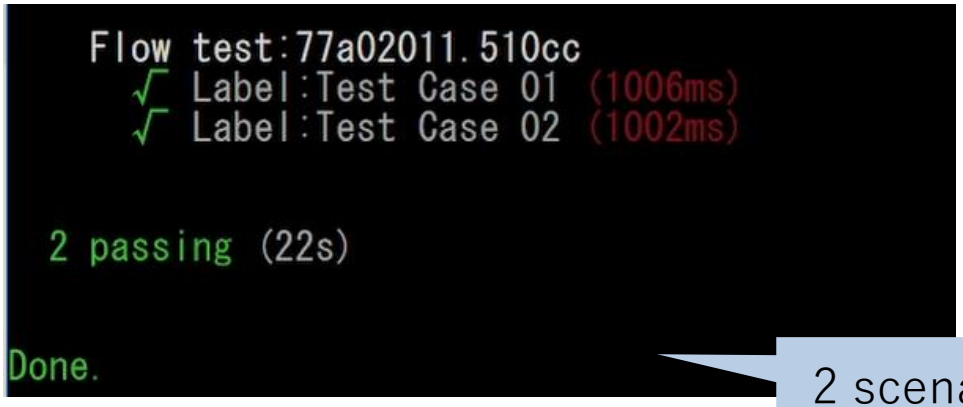
Execution of flow testing from command line

HITACHI
Inspire the Next

```
grunt flow-test --flow="./flow.json"
```

- input
  - --flow:

    JSON file of the flow data
- output



2 scenarios are success

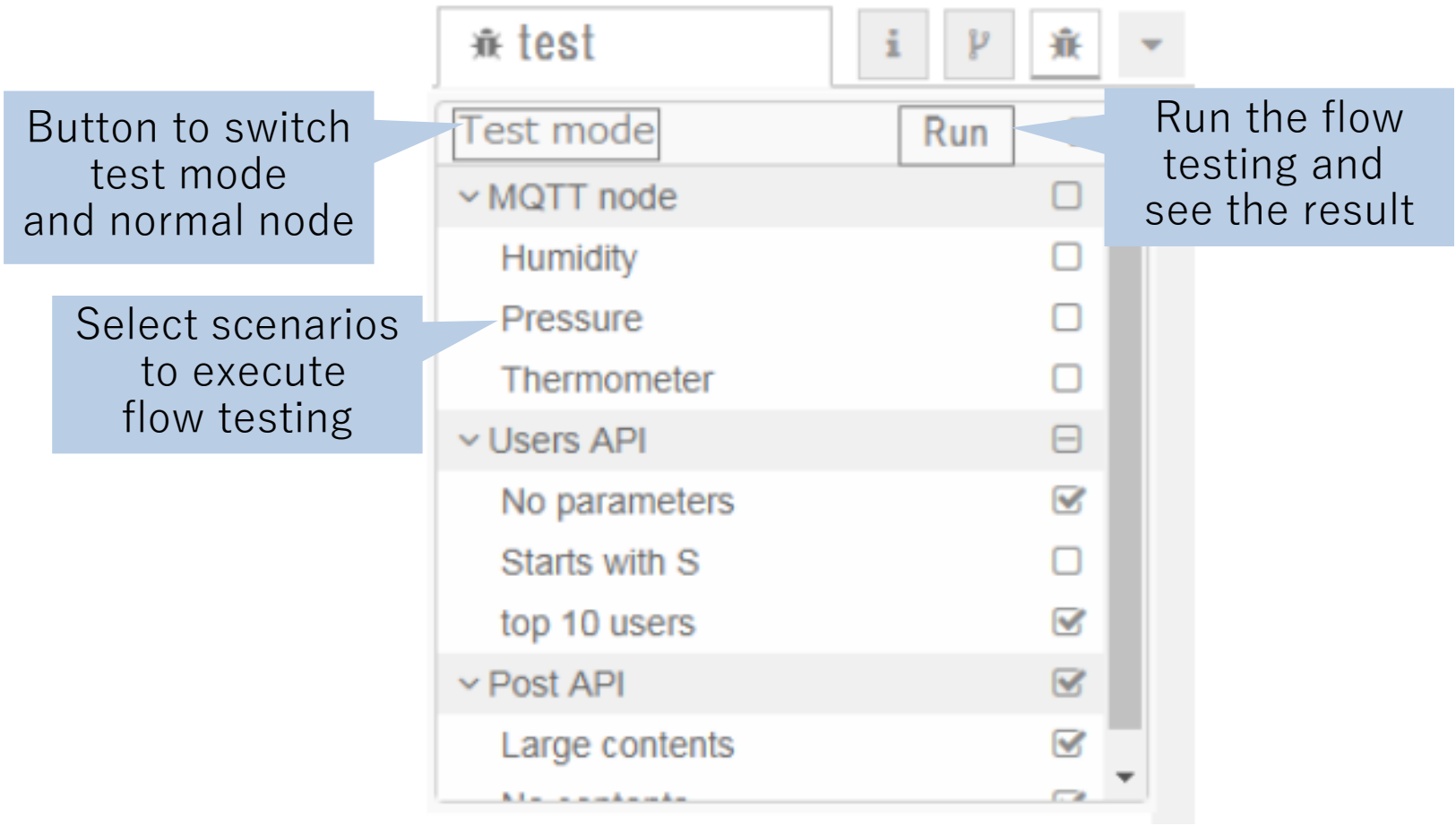
HITACHI
Inspire the Next

Demonstration

Tab UI for flow testing (future functionality)

HITACHI
Inspire the Next

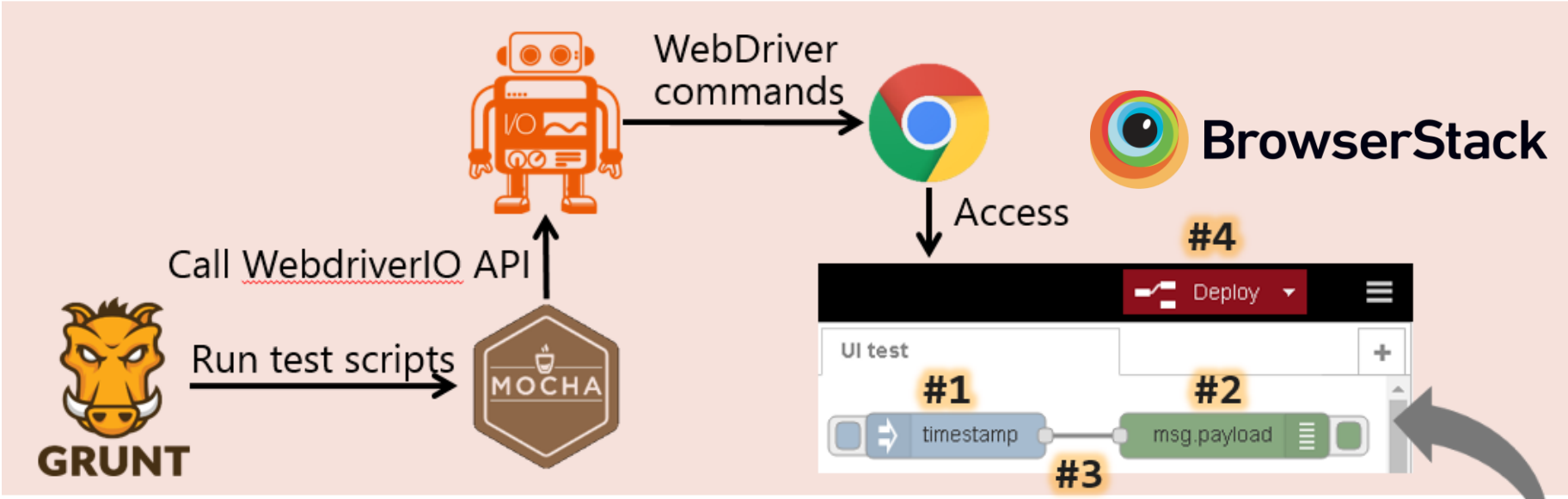
Testing tab on side bar to execute flow testing



Automated UI testing for Node-RED flow editor

HITACHI
Inspire the Next

Automated UI testing to prevent new problems related to the flow editor



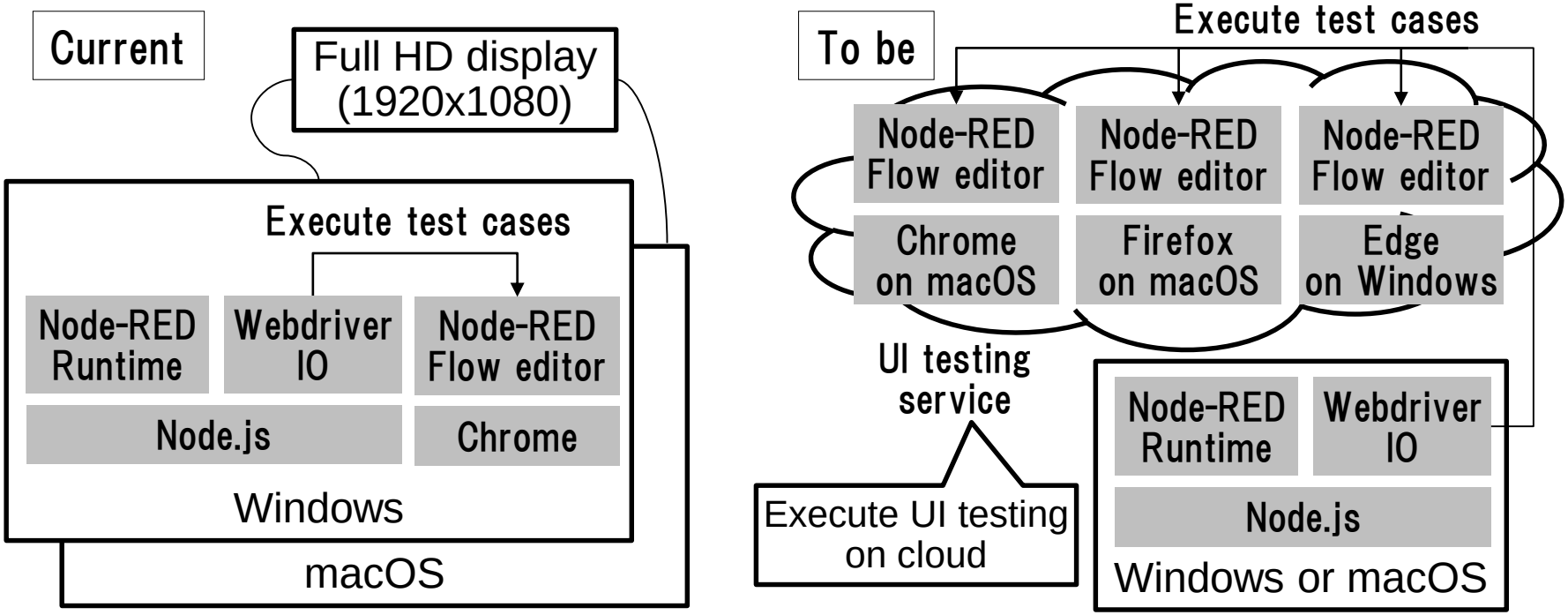

```
// Add an inject node and a debug node
var workspace = require("workspace");
var inject = workspace.addNode("inject");          // #1
var debug  = workspace.addNode("debug", nodeWidth); // #2
// Connect an inject node with a debug node, and deploy it
inject.connect(debug); // #3
workspace.deploy();    // #4
```

Concerns in the current UI testing

HITACHI
Inspire the Next

- In headless mode, it is difficult to check the screen in which UI bug occurred.
- The full HD display is needed when using browser mode (grunt test-ui –headless).
- To test browser testing on Windows and Mac, both environments are required.
- To test browsers like Firefox, additional code to connect to the browser is needed.

-> To solve these problems, I surveyed UI testing services on the cloud and tried to use them.



Comparison of UI testing services

HITACHI
Inspire the Next

We compared UI testing services which WebdriverIO command-line tool (@wdio/cli module) supports in terms of the following options.

- WebSocket support for connections between runtime and editor
- Proxy support for corporate network
- Free plan for OSS

Comparisons of UI testing services

#	Service	WebSocket	Proxy	Free plan for OSS
1	Sauce Labs		√	√
2	BrowserStack	√	√	√
3	Testingbot		√	

We selected BrowserStack

UI testing in BrowserStack

HITACHI
Inspire the Next

We confirmed that all current UI test cases
on Chrome and Firefox in BrowserStack.

[Merits which we found]

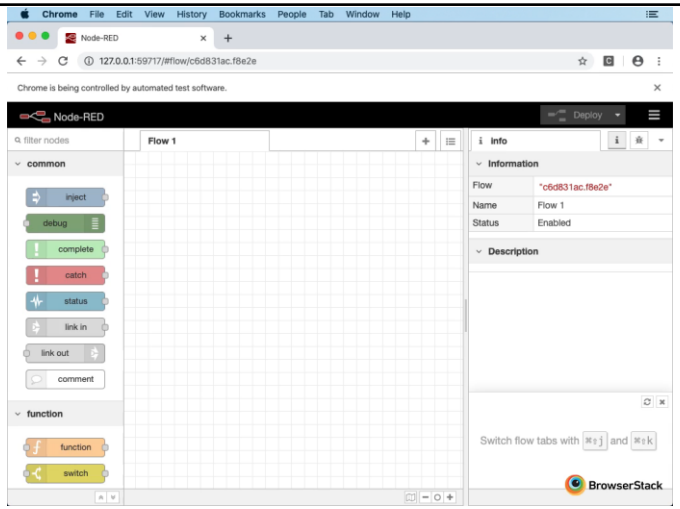
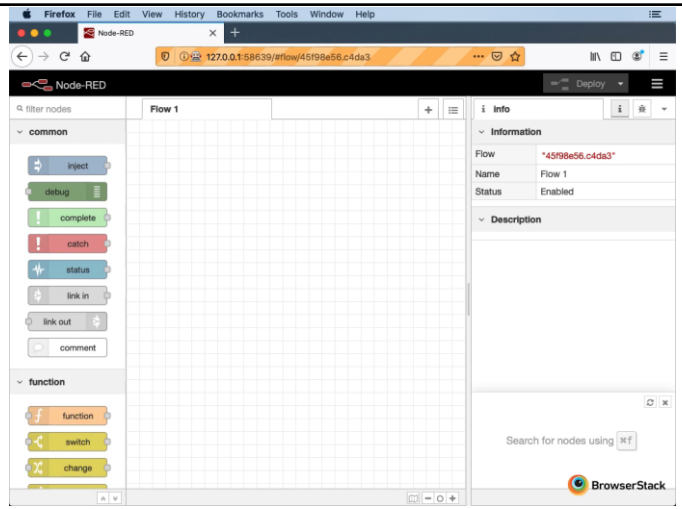
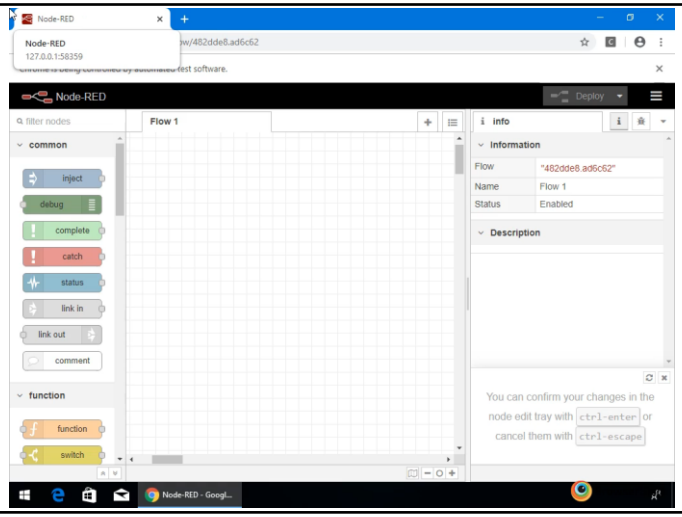
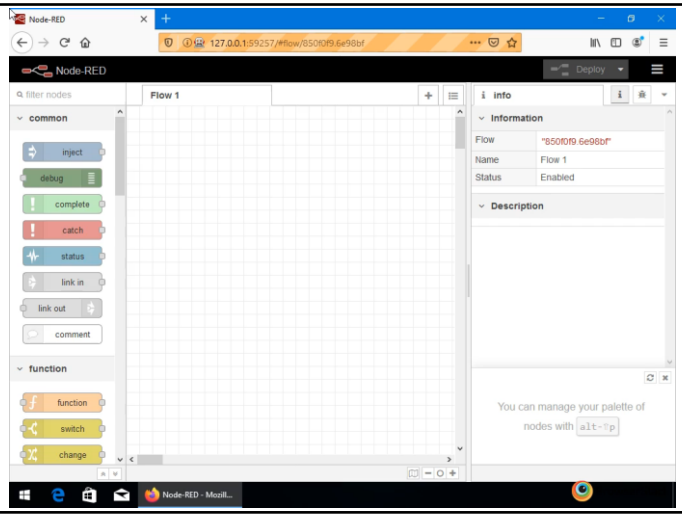
- Supported environment list in each Node-RED release
- Testing the old version of browsers and OS environments
- Testing for the beta version browsers before browser release

#	OS	Chrome			Firefox		
		v80 beta	v79	v78	v73 beta	v72	v71
1	Windows 10	√	√	√	√	√	√
2	Windows 8.1	√	√	√	√	√	√
3	macOS Catalina	√	√	√	√	√	√
4	macOS Mojave	√	√	√	√	√	√

Tested in Node-RED v1.0.3

Demonstration of UI testing on cloud services

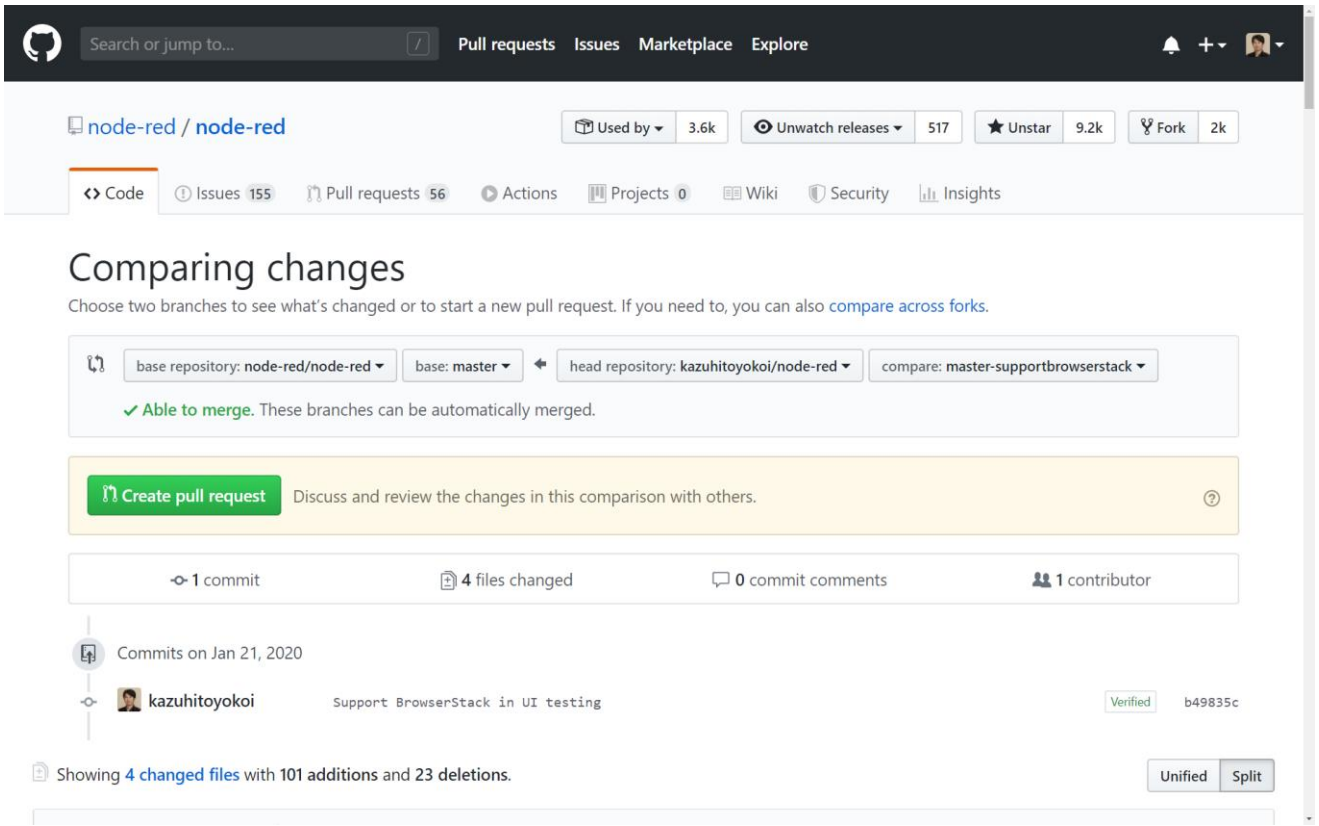
HITACHI
Inspire the Next

	Google Chrome	Firefox
macOS		
Windows		

Pull request to support UI testing on BrowserStack

HITACHI
Inspire the Next

#	Question
1	Could you accept for me to submit the pull request to support the UI testing on BrowserStack?



https://github.com/node-red/node-red/compare/master...kazuhitoyokoi:master-supportbrowserstack