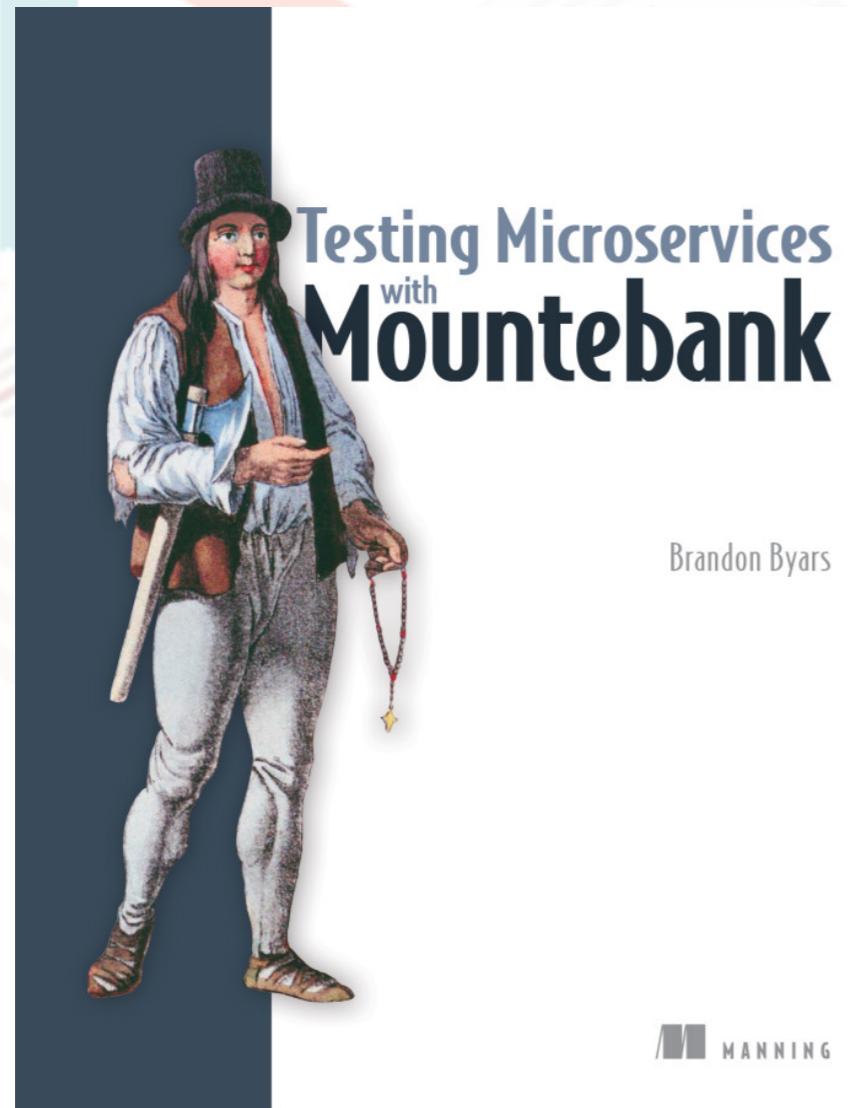


# จำลองระบบเพื่อใช้ในการทดสอบ ด้วย

## Mountebank 101



- Hello, World! Mountebank
- Predicates
- Record/Replay
- Behavior and Programming mountebank
- Adding Behaviors



# Install Mountebank



# Setup Node.js(Windows)

## Downloads

Latest LTS Version: 12.16.1 (includes npm 6.13.4)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.



[Windows Installer \(.msi\)](#)

[Windows Binary \(.zip\)](#)

[macOS Installer \(.pkg\)](#)

[macOS Binary \(.tar.gz\)](#)

[Linux Binaries \(x64\)](#)

[Linux Binaries \(ARM\)](#)

[Source Code](#)

32-bit	64-bit
32-bit	64-bit
64-bit	
64-bit	
64-bit	
ARMv7	ARMv8
<a href="#">node-v12.16.1.tar.gz</a>	

- Goto [Node.js downloads](#) and download Node.js for Windows
  - LTS
  - Current



# Run the Node.js Installer(Windows)

- Welcome to the Node.js setup wizard
  - Select **Next**
- End-User License Agreement (EULA)
  - Check **I accept the terms in the License Agreement**
  - Select **Next**
- Destination Folder
  - Select **Next**
- Custom Setup
  - Select **Next**
- Ready to install Node.js
  - Select **Install**
  - *Note:* This step requires Administrator privileges.
  - If prompted, authenticate as an Administrator.
- Installing Node.js
  - Let the installer run to completion.
- Completed the Node.js Setup Wizard
  - Click **Finish**



# Setup Node.js(Mac)

Install Homebrew

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

Install Node.js

```
brew install node
```



# Verify That Node.js was Properly Installed and Install Mountebank

Run command

```
node -v
```

Update npm

```
npm install npm -g
```

Install mountebank

```
npm install mountebank -g
```



# **Day1: Hello, World! Mountebank**



# The Problem with End-to-End Testing

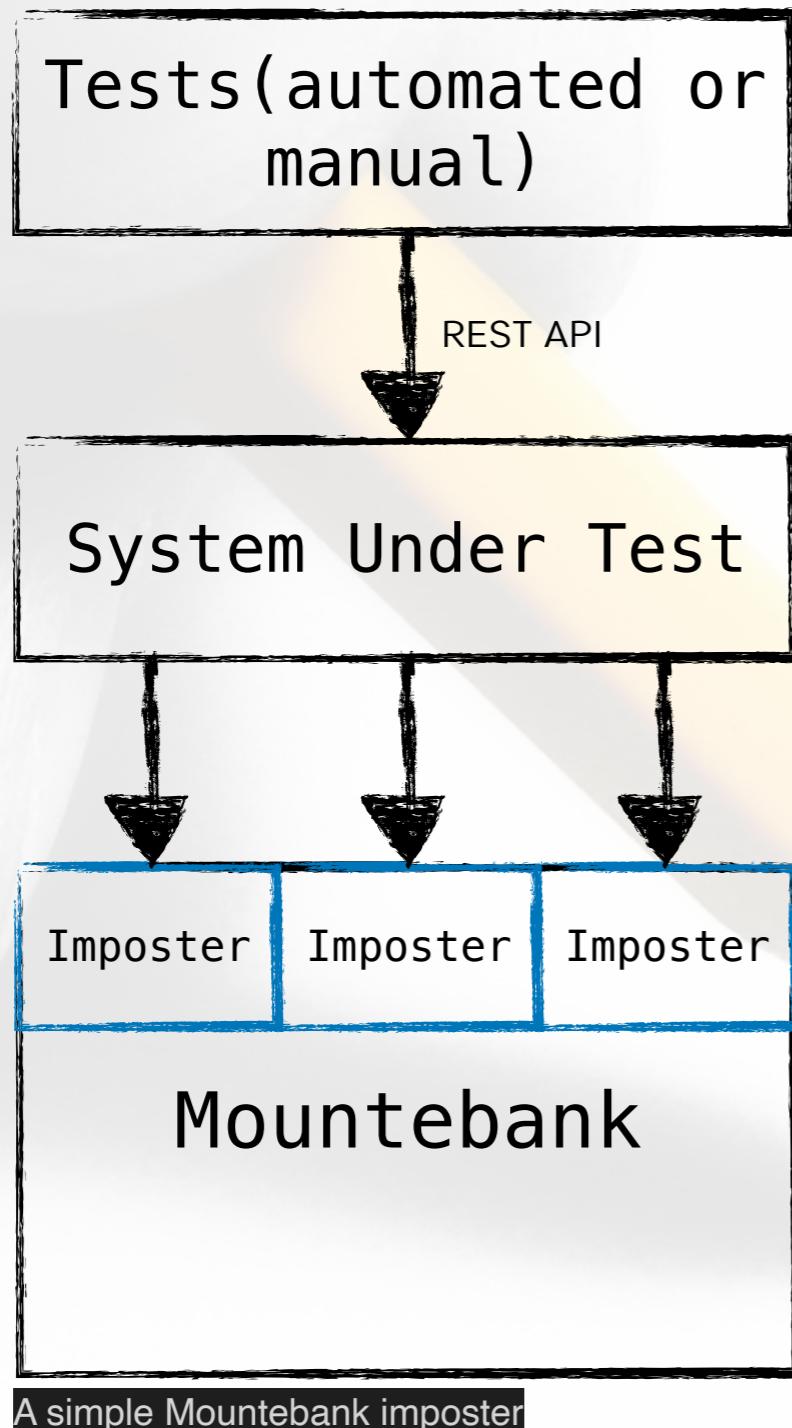
- Time
- Cost
- Slow feedback loop

## 3A pattern: Arrange, Act, Assert, Cleanup

- Setup data test
- Execute the system under test
- Assert the expected response
- Clean the state



# Imposters: virtual services, a lightweight operation



- Mountebank provides configure virtual services, which are called **imposters**.
- Each imposter represents a socket that acts as the virtual service and accepts connections from the real service you are testing.
- Spinning up and shutting down imposters is a **lightweight** operation.

from: Testing Microservice with Mountebank page 17



# How mountebank views an HTTP request

```
POST /products?page=2&itemsPerPage=2 HTTP/1.1  
Host: api.petstore.com  
Content-Type: application/json  
  
{  
  "key": "asdul7890"  
}
```

HTTP request's view

```
{  
  "method": "POST",  
  "path": "/products",  
  "query": {  
    "page": 2,  
    "itemsPrePage": 2  
  },  
  "headers": {  
    "Host": "api.petstore.com",  
    "Content-Type": "application/json"  
  },  
  "body": "{\n    \"key\": \"asdul7890\"\n  }"
```

Mountebank's view

from: Testing Microservice with Mountebank page 26



# How mountebank views an HTTP response

```
HTTP/1.1 200 OK
Date: Sun, 05 Apr 2020 10:10:10 GMT
Content-Type: application/json

{
  "key": "asdul7890"
}
```

HTTP response's view

```
{
  "statusCode": 200,
  "headers": {
    "Date": "Sun, 05 Apr 2020 10:10:10 GMT",
    "Content-Type": "application/json"
  },
  "body": "{\n  \"key\": \"asdul7890\"\n}"
}
```

Mountebank's view

from: Testing Microservice with Mountebank page 27



# Hello, World Mountebank

mountebank - over the wire test doubles

home impostaers logs config Fork me on GitHub Search...

**Welcome, friend**

mountebank is the first open source tool to provide cross-platform, multi-protocol test doubles over the wire. Simply point your application under test to mountebank instead of the real dependency, and test like you would with traditional stubs and mocks.

mountebank is the most capable open source service virtualization tool in existence, and will cure what ails you, guaranteed.

**How it works**

mountebank employs a legion of *imposters* to act as on-demand test doubles. Your test communicates to mountebank over http using the [api](#) to set up [stubs](#), [record and replay proxies](#), and verify [mock expectations](#). In the typical use case, each test will start an imposter during test setup and stop an imposter during test teardown, although you are also welcome to configure mountebank at startup using a [config file](#).

mountebank employs several types of imposters, each responding to a specific protocol. Typically, your test will tell the imposter which port to bind to, and the imposter will open the corresponding socket.

```
graph TD; test -- http --> mountebank; mountebank -- magic --> imposter;
```

View the [getting started guide](#) for a quick introduction.

**Testimonials**

With mountebank, maybe we could have built it in a day.  
- Remus and Romulus, builders of Rome

With mountebank, all the world's a nail.

the apothecary

- getting started
- mental model
- client libraries
- install options
- command line
- security
- faqs
- support

api:

- overview
- contracts
- mock verification
- stubs
  - proxies
  - injection
  - behaviors
  - stub predicates
  - xpath
  - json
  - jsonpath
  - errors

builtin protocols:

- http
- https
- tcp
- smtp

community plugins:

- ldap
- grpc
- create your own

the book:

Create workspace call mb-workspace unix windows

mkdir mb-workspace

Change directory to mb-workspace unix windows

cd mb-workspace

Start Mountebank unix windows

mb

Open web browser then go to <http://localhost:2525>

Stop Mountebank unix windows

command + c

control + c



# Create First Imposter

The screenshot shows the Postman interface with the following details:

- HTTP Method:** POST
- URL:** http://localhost:2525/imposters
- Body (JSON):**

```
1 [ {  
2   "protocol": "http",  
3   "port": 3000,  
4   "stubs": [  
5     {  
6       "responses": [  
7         {  
8           "is": {  
9             "statusCode": 200,  
10            "headers": { "Content-Type": "text/plain" },  
11            "body": "Hello, World!"  
12          }  
13        ]  
14      }  
15    ]  
16  ]  
17 }]
```
- Response (Pretty JSON):**

```
1 [ {  
2   "protocol": "http",  
3   "port": 3000,  
4   "numberOfRequests": 0,  
5   "recordRequests": false,  
6   "requests": [],  
7   "stubs": []  
8 }
```
- Imposters Table:**

	name	protocol	port	# requests
Q X	http:3000	http	3000	0
+ (green)				
- Request:** (Text input field)
- Response:** (Text input field)

HTTP Method

POST

URL

http://localhost:2525/imposters

Body

```
{  
  "protocol": "http",  
  "port": 3000,  
  "stubs": [  
    {  
      "responses": [  
        {  
          "is": {  
            "statusCode": 200,  
            "headers": { "Content-Type": "text/plain" },  
            "body": "Hello, World!"  
          }  
        ]  
      }  
    ]  
  }]
```

Open web browser then go to <http://localhost:2525/imposters>



# Call The First Imposter

DELETE http://localhost:2525/imposters/3000

Params Headers (9) Body Pre-request Script Tests Settings Cookies Code

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (5) Test Results Status: 200 OK Time: 7ms Size: 842 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "protocol": "http",
3   "port": 3000,
4   "numberOfRequests": 1,
5   "recordRequests": false,
6   "requests": [],
7   "stubs": [
8     {
9       "responses": [

```

HTTP Method

DELETE

URL

<http://localhost:2525/imposters/3000>

## Imposters

This page is intended to host a single page app providing a user interface to both explore the mountebank API, and to use mountebank without having to use the API. That's no small effort and unlikely to get prioritized. If you'd like to take on the task, mountebank will offer you his undying gratitude. In the meantime, you may explore other UIs, like the one created by [Don Henton](#).

	name	protocol	port	# requests
<a href="#">+</a>				

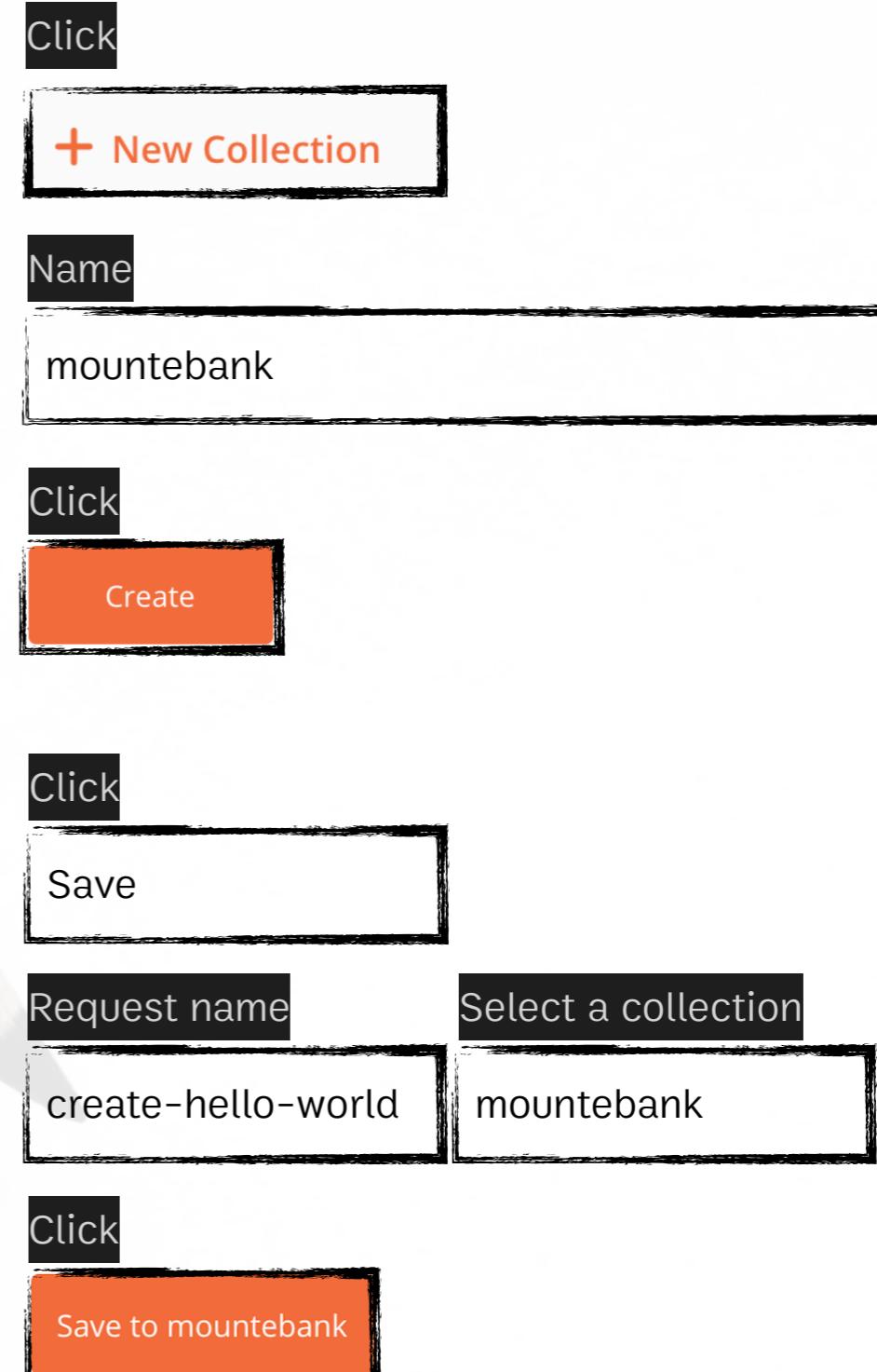
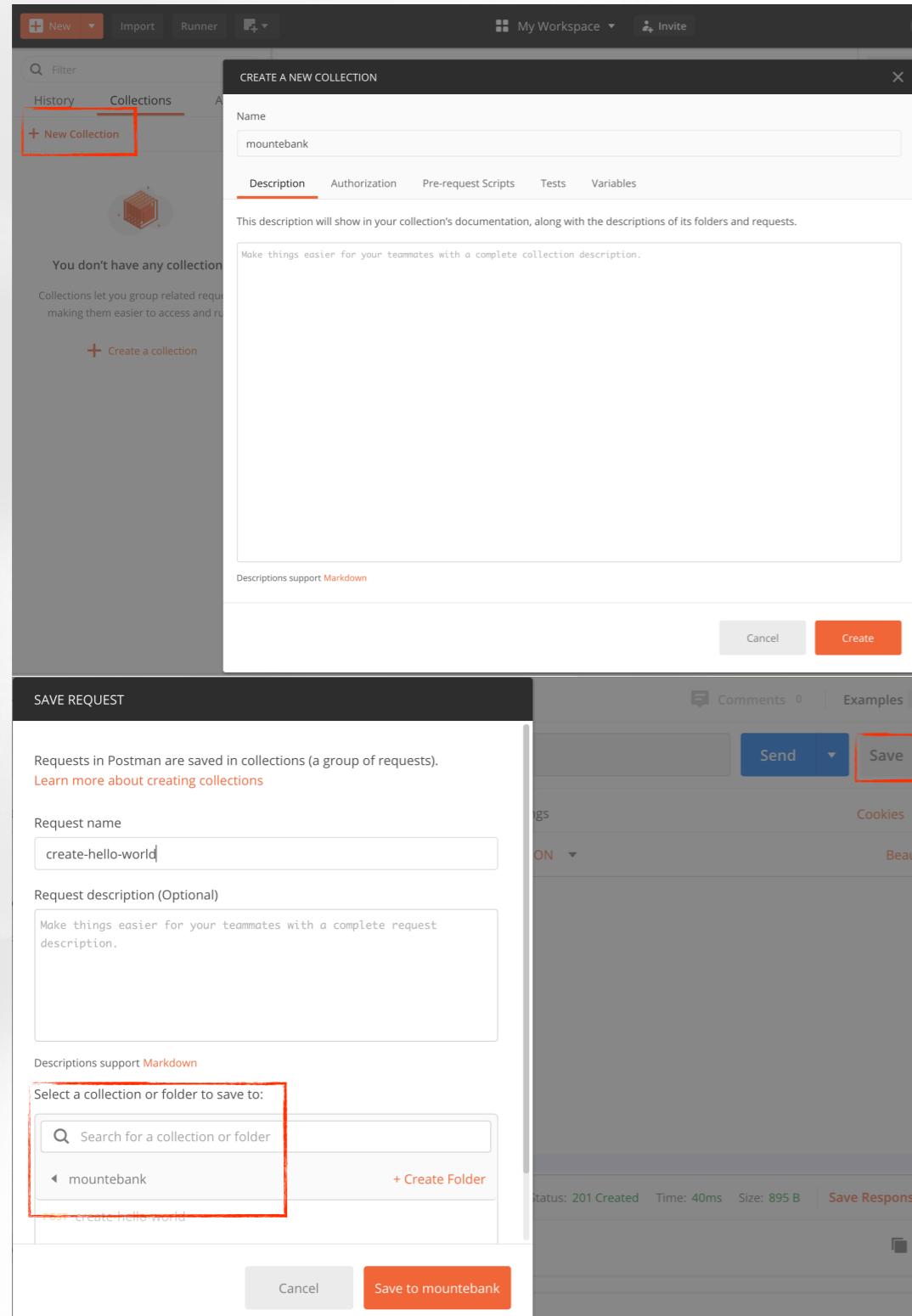
Request:

Response:

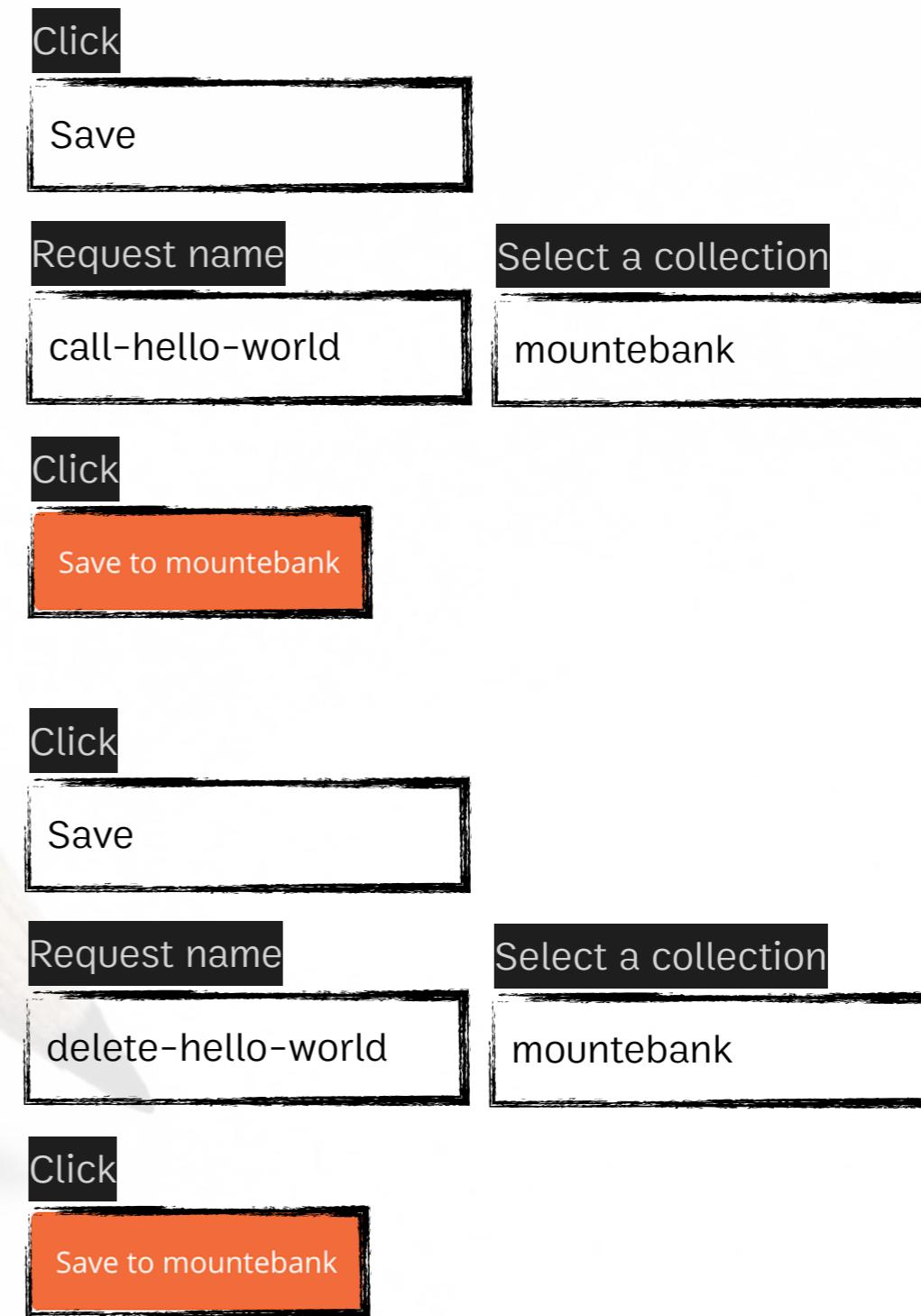
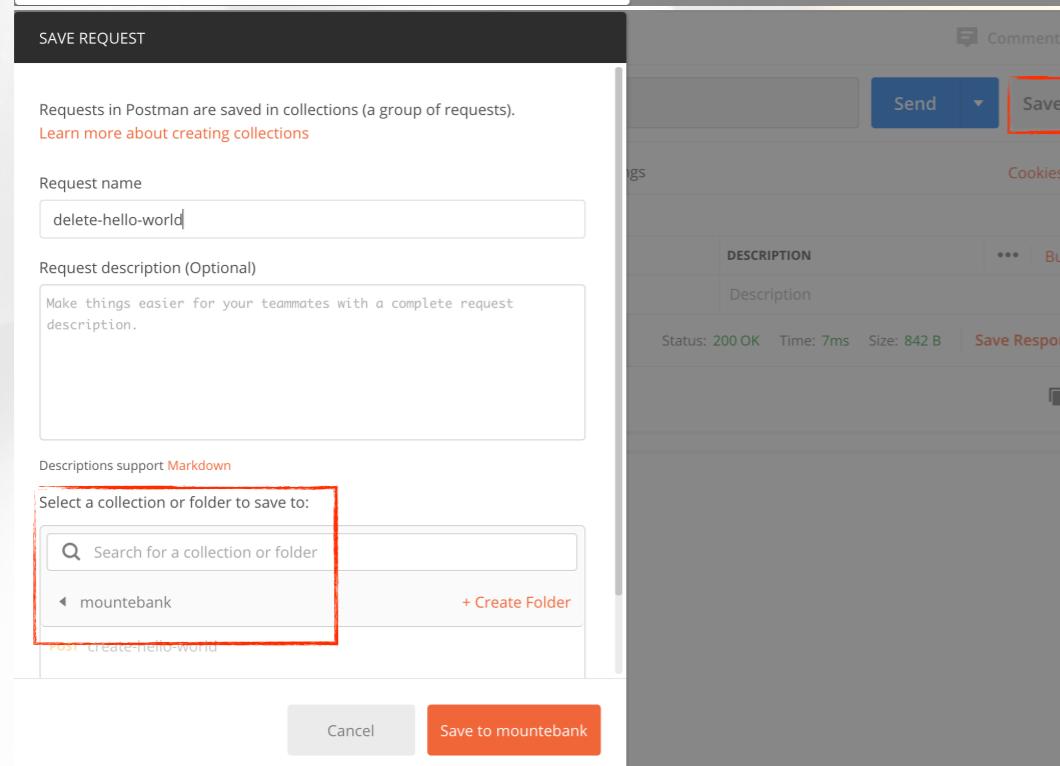
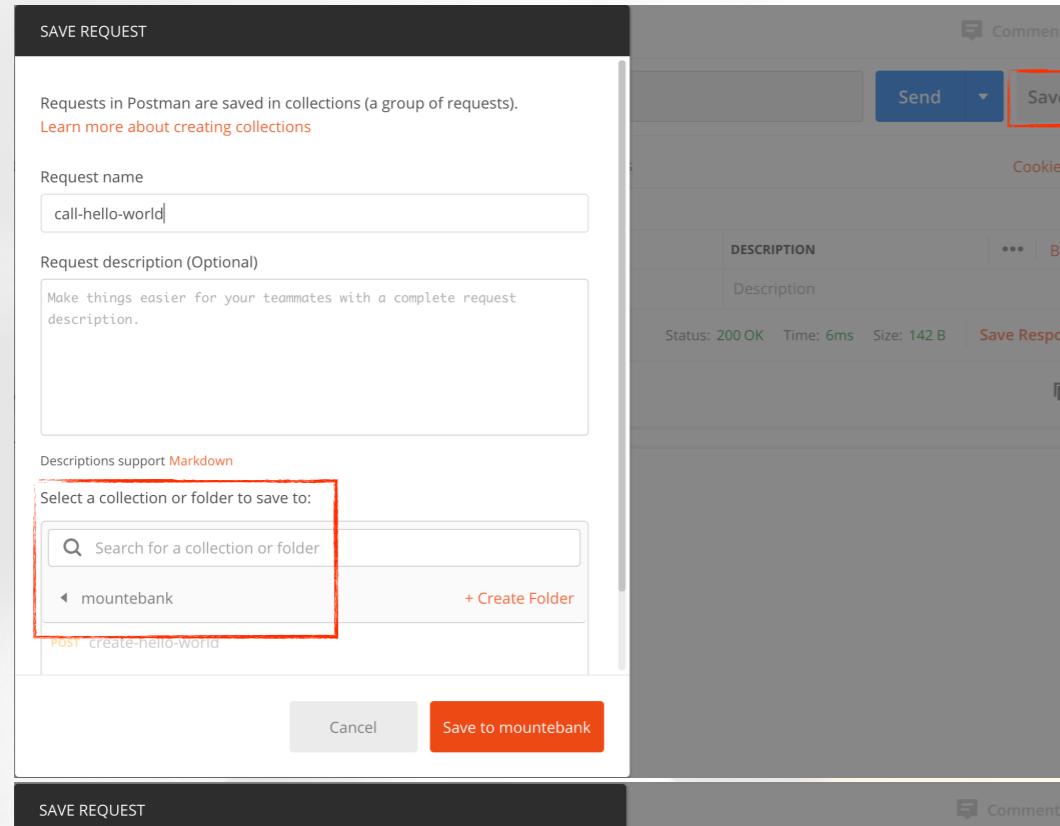
Go to <http://localhost:2525/impostes>



# Save Script to Collection



# Save Script to Collection



# Create Imposters from Config File

## Imposters

This page is intended to host a single page app providing a user interface to both explore the mountebank API, and to use mountebank without having to use the API. That's no small effort and unlikely to get prioritized. If you'd like to take on the task, mountebank will offer you his undying gratitude. In the meantime, you may explore other UIs, like the one created by [Don Henton](#).

	name	protocol	port	# requests
	http:3000	http	3000	0
<a href="#">+</a>				

Request:

Response:

Create file call hello-world.json

```
{  
  "protocol": "http",  
  "port": 3000,  
  "stubs": [  
    {"responses": [  
      {"is": {  
        "statusCode": 200,  
        "headers": { "Content-Type": "text/plain" },  
        "body": "Hello, World!"  
      }  
    }]  
  ]  
}
```

Start Mountebank [unix](#) [windows](#)

```
mb start --configfile hello-world.json
```

Open web browser then go to <http://localhost:2525/impostes>



# The Basic of Mountebank HTTP Response



# The HTTP Response Structure in JSON

The **is** response type, which is the fundamental building block for a stub.

Imposter structure

```
{  
  "protocol": "http",  
  "port": 3000,  
  "stubs": [  
    "responses": [{  
      "is": {  
        "statusCode": 200,  
        "headers": { "Content-Type": "text/plain" },  
        "body": "Hello, World!"  
      }  
    }]  
  }]
```

The HTTP response structure in JSON

```
{  
  "statusCode": 200,  
  "headers": { "Content-Type": "text/plain" },  
  "body": "Hello, World!"  
}
```



# Default Response

The screenshot shows a Postman request configuration. The method is POST, the URL is <http://localhost:2525/imposters>, and the body is a JSON object:

```
1 {  
2   "protocol": "http",  
3   "port": 3001,  
4   "name": "default response",  
5   "defaultResponse": {  
6     "statusCode": 400,  
7     "headers": {  
8       "Connection": "Keep-Alive",  
9       "Content-Length": 0  
10    }  
11  },  
12  "stubs": [{  
13    "responses": [{  
14      "is": { "body": "BOOM!!!" }  
15    }  
16  }]  
17 }
```

The response status is 201 Created, time is 6ms, size is 805 B.

**Imposters**

This page is intended to host a single page app providing a user interface to both explore the mountebank API, and to use mountebank without having to use the API. That's no small effort and unlikely to get prioritized. If you'd like to take on the task, mountebank will offer you his undying gratitude. In the meantime, you may explore other UIs, like the one created by [Don Henton](#).

	name	protocol	port	# requests
Q ✘	http:3000	http	3000	0
Q ✘	http:3001	http	3001	0
+				

Request:

Response:

**HTTP Method**  
POST

**URL**  
<http://localhost:2525/imposters>

**Body**

```
{  
  "protocol": "http",  
  "port": 3001,  
  "name": "default response",  
  "defaultResponse": {  
    "statusCode": 400,  
    "headers": {  
      "Connection": "Keep-Alive",  
      "Content-Length": 0  
    }  
  },  
  "stubs": [  
    "responses": [  
      "is": { "body": "BOOM!!!" }  
    ]  
  ]  
}
```



# Cycling Through Response

The screenshot shows the Postman interface with a POST request to `http://localhost:2525/imposters`. The Body tab is selected, displaying the following JSON:

```
1 {  
2   "protocol": "http",  
3   "port": 3002,  
4   "stubs": [  
5     {  
6       "responses": [  
7         {  
8           "is": { "body": "1" }  
9         },  
10        {  
11          "is": { "body": "2" }  
12        },  
13        {  
14          "is": { "body": "3" }  
15        }  
16      ]  
17    }  
18  ]  
19 }
```

The response status is 201 Created. Below the main interface, there's a summary table titled "Imposters" and a "Request:" field.

	name	protocol	port	# requests
🔍 ✗	http:3000	http	3000	0
🔍 ✗	http:3001	http	3001	0
🔍 ✗	http:3002	http	3002	0

Request:  
Response:

HTTP Method  
POST

URL  
`http://localhost:2525/imposters`

Body

```
{  
  "protocol": "http",  
  "port": 3002,  
  "stubs": [  
    {  
      "responses": [  
        {  
          "is": { "body": "1" }  
        },  
        {  
          "is": { "body": "2" }  
        },  
        {  
          "is": { "body": "3" }  
        }  
      ]  
    }  
  ]  
}
```



# Create Imposters from Config File

Create file call impostaers.json

```
{ "imposters": [ { "protocol": "http", "port": 3000, "stubs": [ { "responses": [ { "is": { "statusCode": 200, "headers": { "Content-Type": "text/plain" }, "body": "Hello, World!" } } ] } ], "protocol": "http", "port": 3001, "name": "Default Response", "defaultResponse": { "statusCode": 400, "headers": { "Connection": "Keep-Alive", "Content-Length": 0 } }, "stubs": [ { "responses": [ { "is": { "body": "BOOM!!!" } } ] } ], "protocol": "http", "port": 3002, "stubs": [ { "responses": [ { "is": { "body": "1" } }, { "is": { "body": "2" } }, { "is": { "body": "3" } } ] } ] } }
```

Start Mountebank **unix windows**

```
mb start --configfile impostaers.json
```

## Imposters

This page is intended to host a single page app providing a user interface to both explore the mountebank API, and to use mountebank without having to use the API. That's no small effort and unlikely to get prioritized. If you'd like to take on the task, mountebank will offer you his undying gratitude. In the meantime, you may explore other UIs, like the one created by [Don Henton](#).

	name	protocol	port	# requests
	<a href="#">http:3000</a>	http	3000	0
	<a href="#">http:3001</a>	http	3001	0
	<a href="#">http:3002</a>	http	3002	0
<a href="#">+</a>				

Request:

Response:



# Create Imposters from Config File with EJS

Create file call impostaers.ejs

```
{  
  "imposters": [  
    <%- include hello-world.json %>  
  ]  
}
```

Start Mountebank **unix windows**

```
mb start --configfile impostaers.ejs
```

file hello-world.json

```
{  
  "protocol": "http",  
  "port": 3000,  
  "stubs": [  
    {"responses": [  
      {"is": {  
        "statusCode": 200,  
        "headers": { "Content-Type": "text/plain" },  
        "body": "Hello, World!"  
      }  
    ]}  
  ]  
}
```

file default-response.json

```
{  
  "protocol": "http",  
  "port": 3001,  
  "name": "default response",  
  "defaultResponse": {  
    "statusCode": 400,  
    "headers": {  
      "Connection": "Keep-Alive",  
      "Content-Length": 0  
    }  
  },  
  "stubs": [  
    {"responses": [  
      {"is": { "body": "BOOM!!!" }}  
    ]}  
  ]  
}
```

file cycling-through-response.json

```
{  
  "protocol": "http",  
  "port": 3002,  
  "stubs": [  
    {"responses": [  
      {"is": { "body": "1" }}  
    ]},  
    {"responses": [  
      {"is": { "body": "2" }}  
    ]},  
    {"responses": [  
      {"is": { "body": "3" }}  
    ]}  
  ]  
}
```



# Create Imposters from Config File with EJS

Create file call impostaers.ejs

```
{  
  "imposters": [  
    <%- include hello-world.json %>,  
    <%- include default-response.json %>,  
    <%- include cycling-through-response.json %>  
  ]  
}
```

Start Mountebank unix windows

```
mb start --configfile impostaers.ejs
```

file hello-world.json

```
{  
  "protocol": "http",  
  "port": 3000,  
  "stubs": [  
    {"responses": [  
      {"is": {  
        "statusCode": 200,  
        "headers": { "Content-Type": "text/plain" },  
        "body": "Hello, World!"  
      }  
    ]}  
  ]  
}
```

file default-response.json

```
{  
  "protocol": "http",  
  "port": 3001,  
  "name": "default response",  
  "defaultResponse": {  
    "statusCode": 400,  
    "headers": {  
      "Connection": "Keep-Alive",  
      "Content-Length": 0  
    }  
  },  
  "stubs": [  
    {"responses": [  
      {"is": { "body": "BOOM!!!" }}  
    ]}  
  ]  
}
```

file cycling-through-response.json

```
{  
  "protocol": "http",  
  "port": 3002,  
  "stubs": [  
    {"responses": [  
      {"is": { "body": "1" }}  
    ]},  
    {"responses": [  
      {"is": { "body": "2" }}  
    ]},  
    {"responses": [  
      {"is": { "body": "3" }}  
    ]}  
  ]  
}
```



# Basic Command and Options

Start Mountebank **unix**

```
mb start --configfile imposta
```

**windows**

```
mb start --configfile imposta
```

Stop Mountebank **unix**

```
mb stop
```

**windows**

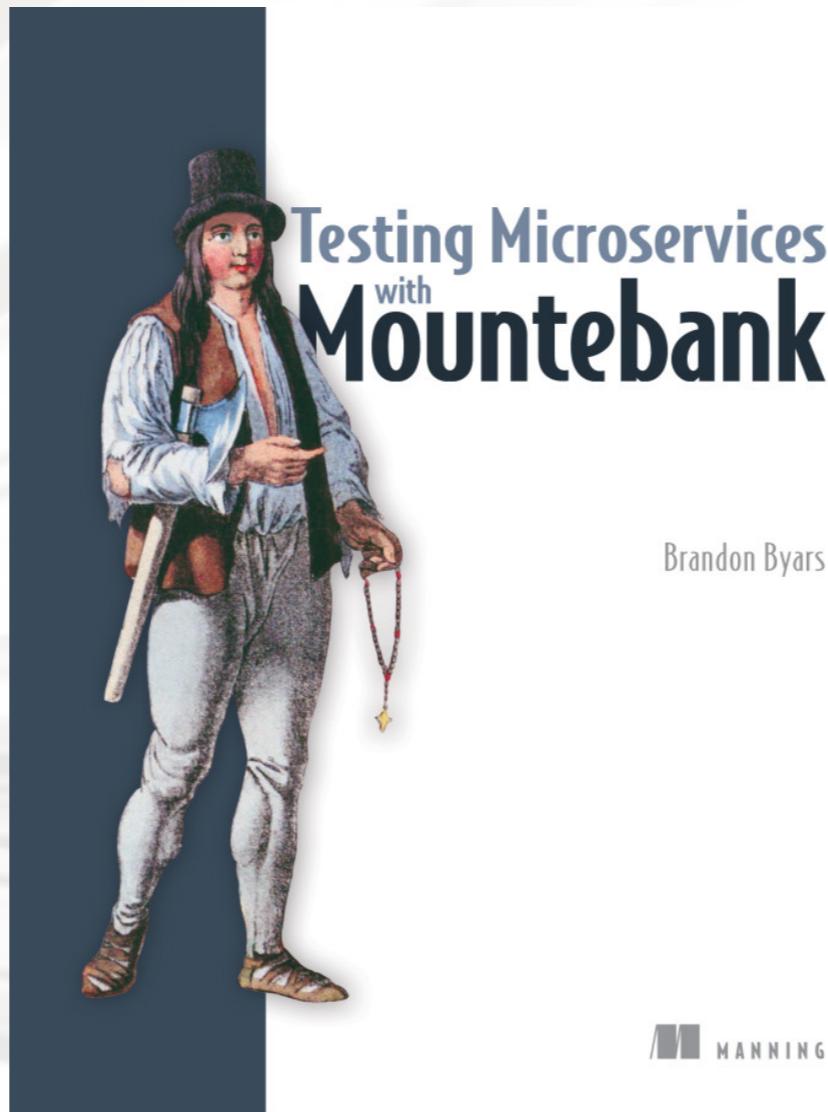
```
control + c
```

Restart Mountebank **unix**

```
mb restart --configfile imposta &
```



# Books to Read and Practice



<https://www.manning.com/books/testing-microservices-with-mountebank>



บริษัท สยามชำนาญกิจ จำกัด และเพื่อนพ้องน้องพี่