



How to instance a KVS-Sandbox Locally



Made at November 12 , 2021 - The purpose of this documentation is to exemplify how to run a local instance of a kvs-service.

Sandbox Services

Sandbox Services is a functionality that attempts to optimize and improve developer's work by enabling them to use different fury services locally on their computers. To this end, real fury services replicas are instantiated in order to allow the developers to connect and use real services instead of mocking their functionalities.

Documentation utils:

- <https://furydocs.io/client/guide/#!/furycli/sandbox/introduction>
- <https://furydocs.io/sandbox-services-docs/guide/#!/>

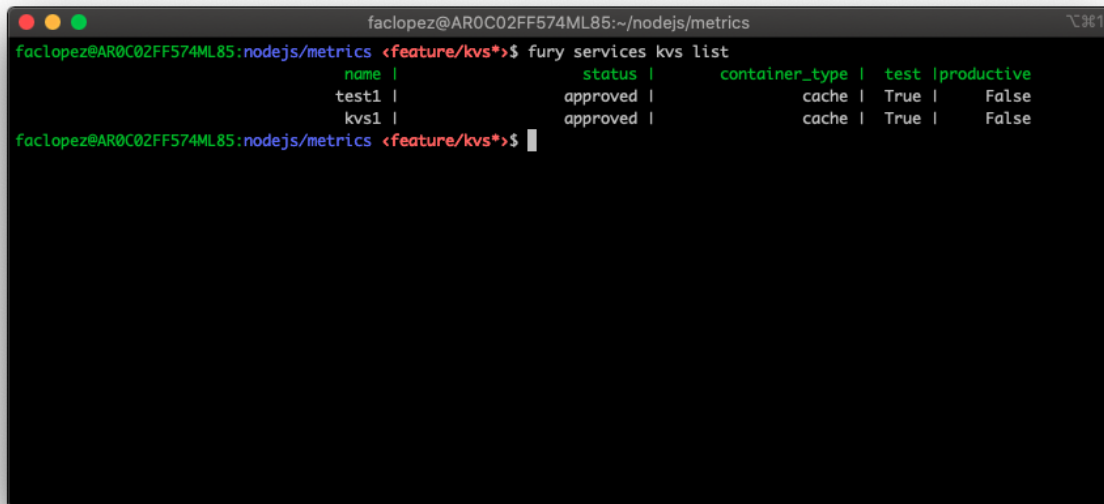
Create the KVS service in fury

For our example we will use a cache type kvs. We will not exemplify this step in detail. For more information see the official documentation in :

https://furydocs.io/kvs-docs/guide/#!/kvs_userManual?id=_1-creación-de-un-kvs

Once our application is downloaded to local and we are inside the work folder, we list the list of KVS services that we have. In our example, the application is called **metrics**.

```
fury services kvs list
```



```
faclopez@AR0C02FF574ML85:~/nodejs/metrics$ fury services kvs list
  name |          status | container_type | test | productive
-----|-----|-----|-----|-----
 test1 |      approved |         cache |  True |        False
  kvs1 |      approved |         cache |  True |        False
faclopez@AR0C02FF574ML85:~/nodejs/metrics <feature/kvs*>$
```

We have two KVS created in this project, so we are going to connect locally the **test1** KVS.

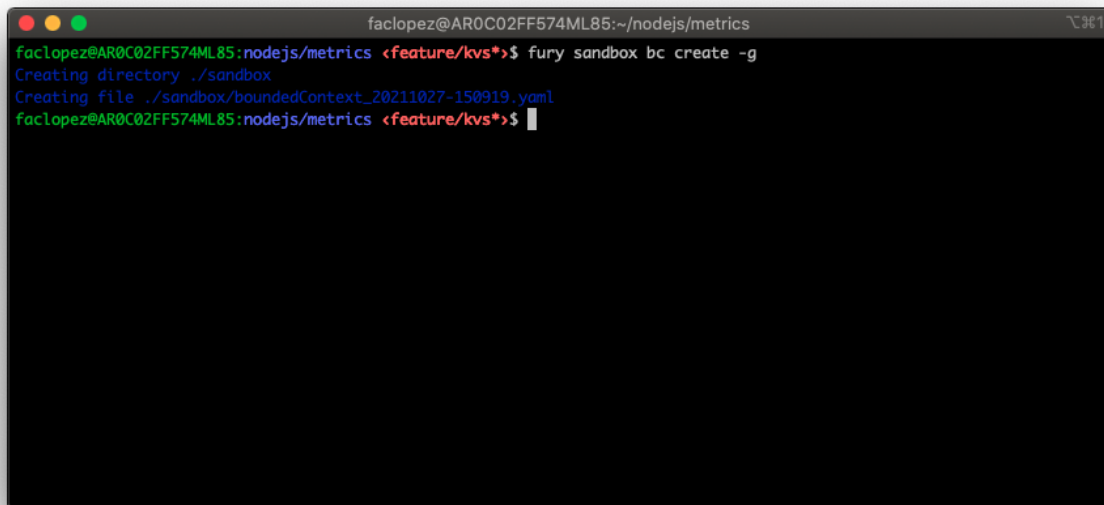
How we do it? follow me...

Creating a Bounded Context

A Bounded Context is a group of services specifications that allows the Sandbox Service to spin up a set of services that will work together. It is essentially like a recipe that specifies what services you want to have available once your Sandbox Services are created.

The following command generates a YAML file with the template needed to create a new BC in the path `./sandbox` inside your application folder.

```
fury sandbox bc create -g
```

A terminal window screenshot showing the execution of the command 'fury sandbox bc create -g'. The terminal output shows the command being executed, the directory './sandbox' being created, and a file './sandbox/boundedContext_20211027-150919.yaml' being created. The prompt then returns to the user.

```
faclopez@AR0C02FF574ML85:~/nodejs/metrics$ fury sandbox bc create -g
Creating directory ./sandbox
Creating file ./sandbox/boundedContext_20211027-150919.yaml
faclopez@AR0C02FF574ML85:~/nodejs/metrics$
```

The .yaml file will look like this:

```
name: name
description: description
services: []
version: 1.0.0-test
```

We only need to change the name, description and services for our Bounded Context [BC]. The file should be in this case:

```
name: testBC
description: our first bounded context
services: [test1]
version: 1.0.0-test
```

IMPORTANT! Keep in mind that test1 is the name of the KVS that we want to have in our BC

Once we have it, we run the following, where `boundedContext_20211027-150919` is the name of our .yaml file

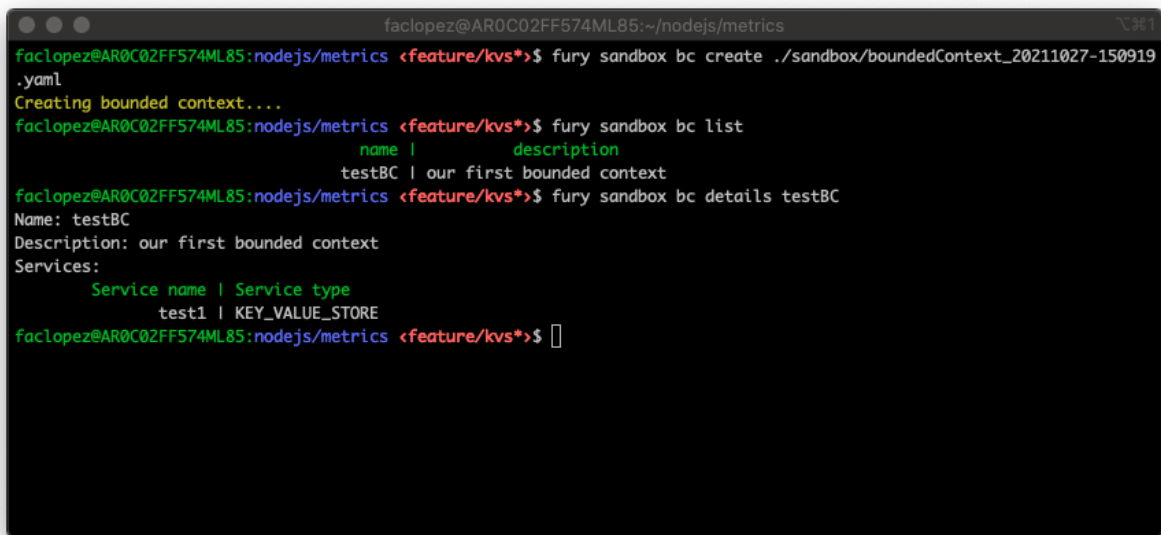
```
fury sandbox bc create ./sandbox/boundedContext_20211027-150919.yaml
```

To be sure if it works...

```
fury sandbox bc list
```

Now to see the details...

```
fury sandbox bc details testBC
```

A terminal window with a dark background and light-colored text. The window title is 'faclopez@AR0C02FF574ML85:~/nodejs/metrics'. The terminal shows the following commands and output:

```
faclopez@AR0C02FF574ML85:~/nodejs/metrics <feature/kvs*>$ fury sandbox bc create ./sandbox/boundedContext_20211027-150919.yaml
Creating bounded context....
faclopez@AR0C02FF574ML85:~/nodejs/metrics <feature/kvs*>$ fury sandbox bc list
      name |      description
      testBC | our first bounded context
faclopez@AR0C02FF574ML85:~/nodejs/metrics <feature/kvs*>$ fury sandbox bc details testBC
Name: testBC
Description: our first bounded context
Services:
  Service name | Service type
      test1 | KEY_VALUE_STORE
faclopez@AR0C02FF574ML85:~/nodejs/metrics <feature/kvs*>$
```

We proudly see that we have created the BC with the associate KVS.

Instance the Bounded Context

To create an instance of the BC we do...

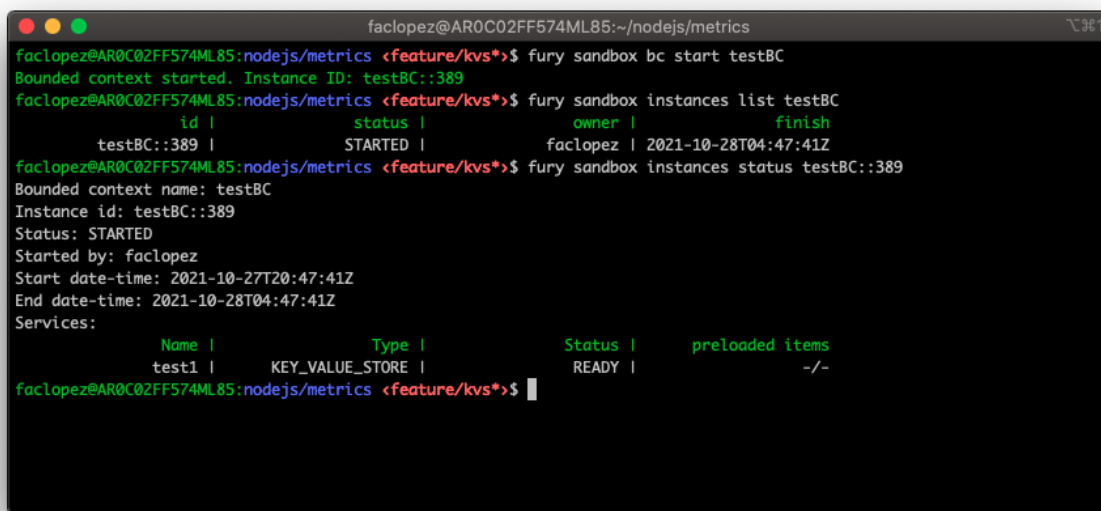
```
fury sandbox bc start testBC
```

To list the instances of the BC...

```
fury sandbox instances list testBC
```

To see the details...

```
fury sandbox instances status testBC::389
```



```
faclopez@AR0C02FF574ML85:~/nodejs/metrics
faclopez@AR0C02FF574ML85:nodejs/metrics <feature/kvs*>$ fury sandbox bc start testBC
Bounded context started. Instance ID: testBC::389
faclopez@AR0C02FF574ML85:nodejs/metrics <feature/kvs*>$ fury sandbox instances list testBC
      id |          status |          owner |          finish
testBC::389 |          STARTED |          faclopez | 2021-10-28T04:47:41Z
faclopez@AR0C02FF574ML85:nodejs/metrics <feature/kvs*>$ fury sandbox instances status testBC::389
Bounded context name: testBC
Instance id: testBC::389
Status: STARTED
Started by: faclopez
Start date-time: 2021-10-27T20:47:41Z
End date-time: 2021-10-28T04:47:41Z
Services:
      Name |          Type |          Status |          preloaded items
test1 | KEY_VALUE_STORE |          READY |          -/-
faclopez@AR0C02FF574ML85:nodejs/metrics <feature/kvs*>$
```

Great, we have our instance on the move.

Set the environment variables

In order to execute locally our sandbox connection, we need to set in our IDE a couple of environment variables. Where and how do we get those variables?

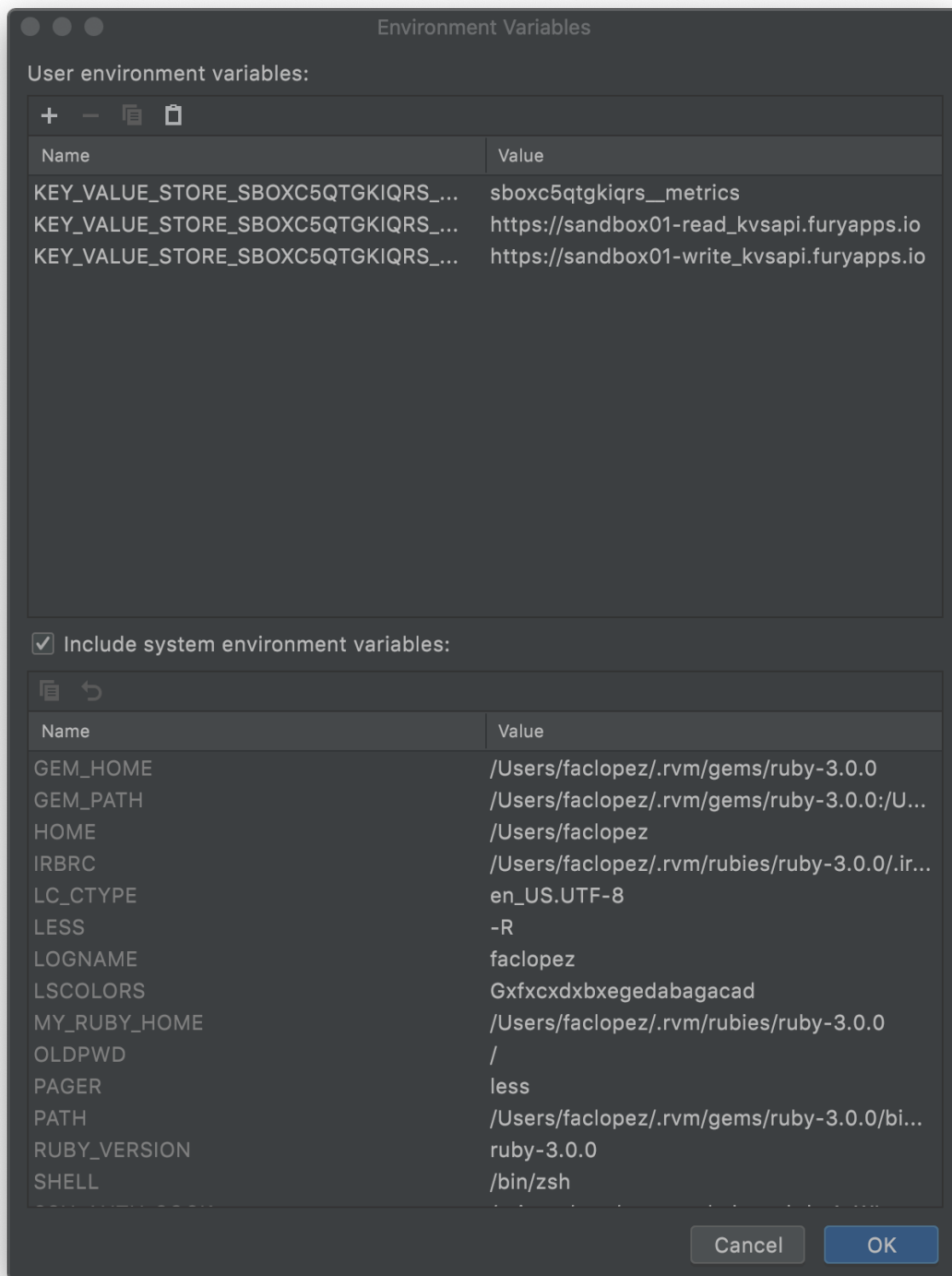
```
fury sandbox instances get-configs testBC::389
```

```
faclopez@AR0C02FF574ML85:~/nodejs/metrics
faclopez@AR0C02FF574ML85:~/nodejs/metrics <feature/kvs*>$ fury sandbox instances get-configs testBC::389
Original service name: test1
Sandbox service name: sbbox5qtgkiqrs
ENV_KEY_VALUE_STORE_SB0XC5QTGKIQRS_CONTAINER_NAME: sbbox5qtgkiqrs__metrics
ENV_KEY_VALUE_STORE_SB0XC5QTGKIQRS_END_POINT_READ: https://sandbox01-read_kvsapi.furyapps.io
ENV_KEY_VALUE_STORE_SB0XC5QTGKIQRS_END_POINT_WRITE: https://sandbox01-write_kvsapi.furyapps.io

faclopez@AR0C02FF574ML85:~/nodejs/metrics <feature/kvs*>$
```

IMPORTANT! take notes of all the variables, including the **Sandbox service name**

Now we set the ENV Variables in our IDE. In this example we will use WebStorm from JetBrains.



If we need to use it with `fury-run` we must add the ENV Variables into the Dockerfile.

```
Dockerfile x
1 FROM hub.furycloud.io/mercadoLibre/nodejs:14-mini
2
3 ENV SCOPE="test"
4 ENV KEY_VALUE_STORE_SBOXC5QTGKIQRS_CONTAINER_NAME=sboxc5qtgkiqrs__metrics
5 ENV KEY_VALUE_STORE_SBOXC5QTGKIQRS_END_POINT_READ=https://sandbox01-read_kvsapi.furyapps.io
6 ENV KEY_VALUE_STORE_SBOXC5QTGKIQRS_END_POINT_WRITE=https://sandbox01-write_kvsapi.furyapps.io
7
```

If we decide to use VsCode, the steps are slightly different...

As our point application runs in NodeJs, we are going to do

```
npm install dotenv
```

And we are going to require the package as early as possible in the main code with

```
require('dotenv').config()
```

Finally we need to create a `.env` file in the project root and set the ENV variables

```
.env U x
.env
1 SCOPE="test"
2 KEY_VALUE_STORE_SBOXC5QTGKIQRS_CONTAINER_NAME=sboxc5qtgkiqrs__metrics
3 KEY_VALUE_STORE_SBOXC5QTGKIQRS_END_POINT_READ=https://sandbox01-read_kvsapi.furyapps.io
4 KEY_VALUE_STORE_SBOXC5QTGKIQRS_END_POINT_WRITE=https://sandbox01-write_kvsapi.furyapps.io
```

Connect the Code with the Sandbox-KVS

Here comes the magic part, how can I tell my code that it has to consume this new local KVS?

Well, the snippet we see when we create our KVS called **test1** contains all the config params. Zooming in, the part when we call our KVS by his name, in NodeJs, looks something like this:

```
const kvsmodule = require('kvsclient');

const Item = kvsmodule.Item,
      Items = kvsmodule.Items,
      KvsClient = kvsmodule.KvsClient,
      ContainerKvsClient = kvsmodule.ContainerKvsClient,
      KvsApiConfiguration = kvsmodule.KvsApiConfiguration;

var kvsclient = new KvsClient({
  kvsApiConfiguration: new KvsApiConfiguration ({
    readTimeout: 2000,
    writeTimeout: 2000
  })
});

var client = new ContainerKvsClient({
  container: 'TEST1',
  delegate: kvsclient
});
```

In our code we are going to substitute `container: 'TEST1'` for `container: "sboxc5qtgkiqrs"`

This `sboxc5qtgkiqrs` is the **Sandbox service name** . That will be all...

Run you project locally

- To run the project you will need to be connected to the VPN.
- A request example could be: <http://localhost:8080/save> and that route purpose is to save an item into the KVS.
- No need to use X-Auth-Token header on the request.

Once we finish our development, we can stop the instances and delete the BC (or not). Those commands are in the fury documentation that we provided at the beginning of this document.

I hope you enjoyed this friendly little tutorial so you don't have to deploy a thousand versions of code to make a query work.