Getting Started

Table of Contents

To make it easier to see what kubernetes resources are being created as you create functions and flows lets use a separate namespace called `funky`.

```
kubectl create namespace funky
kubectl config set-context `kubectl config current-context` --namespace=funky
```

Now we'll install the runtime platform (the controller and the nodejs runtime) and a couple of connectors into the `funky` namespace

```
funktion install platform
funktion install connector http4 timer twitter
```

Create a function

You can create a function from an existing source file. e.g. lets clone an example project:

```
git clone https://github.com/funktionio/funktion-nodejs-quickstart.git
cd funktion-nodejs-quickstart
```

Now we can create a function from a source file as follows:

```
funktion create fn -f src/hello.js
```

Use `apply` instead of `create` so that you can insert or update resources. You can also omit the resource kind `fn` or `flow` and it'll figure out the right kind:

```
funktion apply -f src/hello.js
```

Or you can specify the source code on the command line:

```
funktion create fn -n hello -s 'module.exports = function(context, callback) {
callback(200, "Hello, world!\n"); }'
```

Any of the above options will create a function resource. You can view it via

```
funktion get fn
```

If you wish to keep editing the source code of the function in your editor and have funktion automatically update the running function use the `-w` argument to watch the source file(s):

```
funktion apply -f src/hello.js -w
```

If you have a folder with multiple function source files inside you can pass the directory name or a wildcard pattern:

```
funktion apply fn -f src -w
```

To be able to find the URL of the running function you can type:

```
funktion get fn
```

Or to script it this command is useful:

```
funktion url fn hello
```

which will output the URL to access your function. Or to open it in a browser:

```
funktion url fn hello -o
```

Create a flow

```
funktion create flow timer://bar?period=5000 http://hello/
```

You should now have created a flow. You can view the flow via:

```
funktion get flow
```

To view the output of the flow you can use the following:

```
funktion logs flow timer-bar1
```

You should eventually see the output of the timer events triggering your `hello` function.

To delete the flow:

```
funktion delete flow timer-bar1
```

Use an existing HTTP endpoint

Flows can work with any endpoints whether they are defined via a `function` or not.

e.g. this flow will use an existing endpoint on the internet

```
funktion create flow timer://bar?period=5000 http://ip.jsontest.com/
```

A more complex example

To see a more real world style example check out the blog splitting and counting example using functions and flows