Kumori PaaS - Getting Started

Kumori Systems v1.0.2, February 2019

Table of Contents

This document provides a quick demonstration of how to try Kumori PaaS platform and deploy your first **Hello-World** application in less than 5 minutes.

Kumori PaaS lets you create and deploy scalable services, managing the complexity of their life cycle, letting you focus on their functionality.

Building complex services is simple when following our service model. Our tools let you get started quickly.

1. A simple example

Before working with Kumori PaaS you need to sign up (if you have not done so already).

After signing up you will be able to interact with the platform and manage your services through its Dashboard.

We also offer a command-line interface tool, that you will be using in what follows.

1.1. Prerequisites

You will need docker installed in your machine.

As stated in the Docker installation guide, to use Docker as a non-root user, you should add your user to the "docker" group with:
`sudo usermod -aG docker $USER`

You also need nodejs 8.9 or later (with npm) to install and run the tools.

Then you can install Kumori CLI as follows:

```
sudo npm install -g @kumori/cli
```

1.2. Simple express server

Create a directory for your project, and go into it:

```
mkdir simpleweb
cd simpleweb
```

Initialize the project workspace:

```
kumori init
```

Kumori's naming scheme for services and components uses a domain name belonging to the user to avoid name collisions. You should now configure that domain name with the CLI for your project:

```
kumori set domain <your_domain>
```

Now we are ready to start building our service. Our simple service will have just one component (the web server). We offer a set of templates (see the Kumori Generator repository for a complete templates list) which you can use to quickly implement simple service patterns. We make use of one of them:

```
kumori project add -t @kumori/workspace:project-hello-world <name>
```

This populates your workspace with the Hello World example elements including:

- A component skeleton within the `components/<your_domain>/<name>` directory. Note that the internal name this component will receive is `eslap://<your_domain>/component/<name>/0_0_1` (further details can be found in the Building Services for the Kumori PaaS guide).
- A service in `services/<your_domain>/<name>` folder. Also note that the internal name of the created service app is `eslap://<your_domain>/service/<name>/0_0_1`.

For this example, we can leave the component and service as it is. However, before deploying our service, we need to build its component. Building a component basically implies packing it together with its dependencies.

```
kumori component build <name>
```

Before we can actually deploy, we need to configure our workspace with a valid API access token. This token can be obtained from Kumori Dashboard - Settings page (three-dot menu at the top right).

With the token in our hands we use:

```
kumori stamp update -t <token> baco
```

`baco` is the current production version of Kumori PaaS.

And now we can deploy like so:

```
kumori deployment deploy <name> -i
```

Which will return us an URL to browse the service we just deployed and a couple of URNs. The first URN is our service identifier. The second URN is the id of the inbound linked to our service. The inbound is used to expose our service using a randomly generated URL and is created when the `-i` flag is used.

If your service responds with a 500 error code, please wait a few minutes and try again. NOTE: An HTTP inbound is a special service used to enable HTTP external access to other services. The inbounds must be paired to a domain which can be either generated automatically or manually set. NOTE: You can deploy your service as many times as you want. Each time, a new and independent instance of your service will be created.

To undeploy your service just run:

```
kumori deployment undeploy <service-urn>
kumori deployment undeploy <inbound-urn>
```

To learn more, continue reading the Quick Start guide you can find in this same repo.