JHipster

Doing microservices with JHipster

Microservices vs Monolithic architecture

The first question JHipster will ask you is the kind of application you want to generate. You have the choice between two architecture styles:

* A “monolithic” architecture uses a single, one-size-fits-all application, which contains both the front-end Angular code, and the back-end Spring Boot code.
* A “microservices” architecture splits the front-end and the back-end, so that it’s easier for your application to scale and survive infrastructure issues.

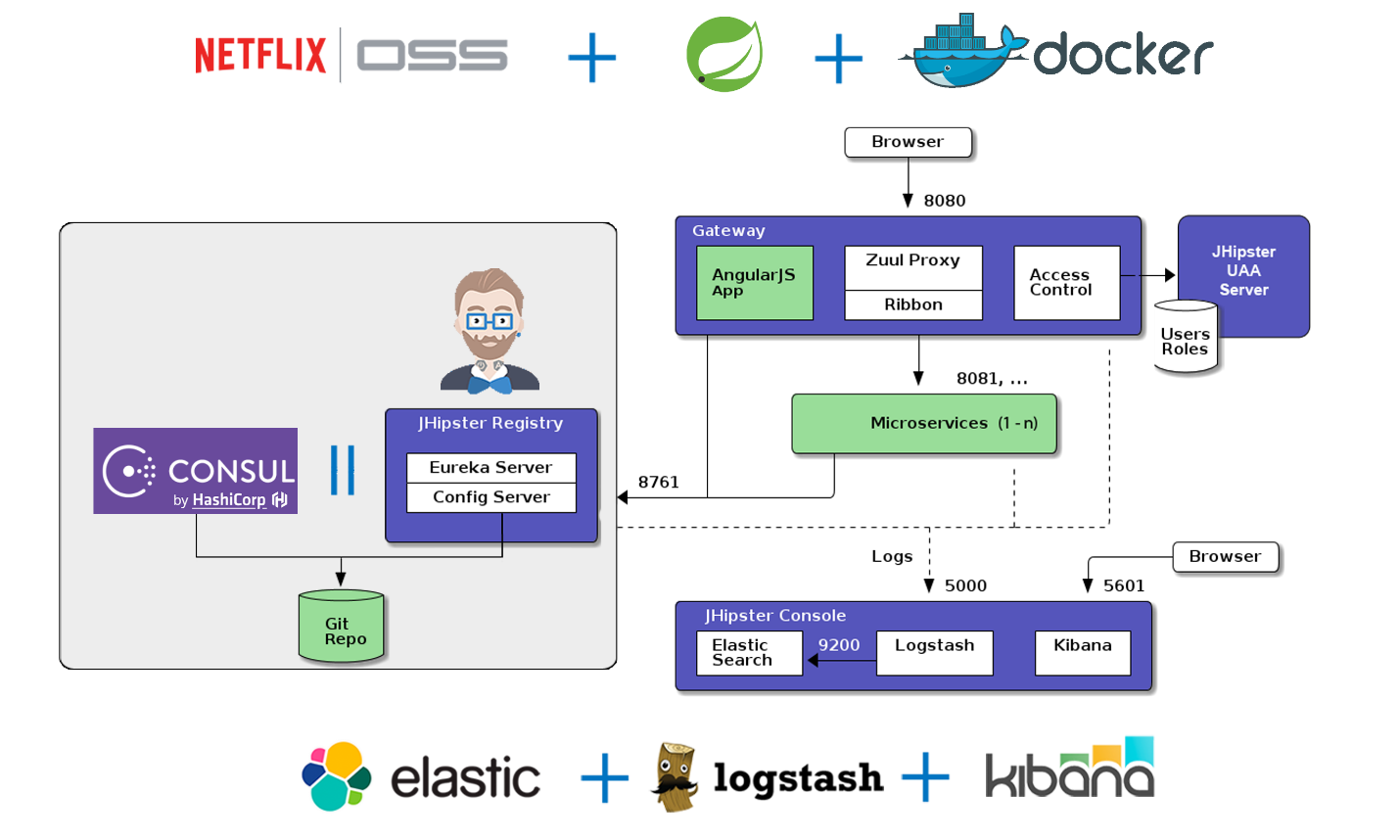
A “monolithic” application is much easier to work on, so if you don’t have any specific requirements, this is the option we recommend, and our default option.

Microservices architecture overview

The JHipster microservices architecture works in the following way:

* A [gateway](https://www.jhipster.tech/api-gateway/) is a JHipster-generated application (using application type microservice gateway when you generate it) that handles Web traffic, and serves an Angular application. There can be several different gateways, if you want to follow the [Backends for Frontends pattern](https://www.thoughtworks.com/insights/blog/bff-soundcloud), but that’s not mandatory.
* [Traefik](https://www.jhipster.tech/traefik/) is a modern HTTP reverse proxy and load balancer that can work with a gateway.
* The [JHipster Registry](https://www.jhipster.tech/jhipster-registry/) is a runtime application on which all applications registers and get their configuration from. It also provides runtime monitoring dashboards.
* [Consul](https://www.jhipster.tech/consul/) is a service discovery service, as well as a key/value store. It can be used as an alternative to the JHipster Registry.
* [JHipster UAA](https://www.jhipster.tech/using-uaa/) is a JHipster-based User Authentication and Authorization system, which uses the OAuth2 protocol.
* [Microservices](https://www.jhipster.tech/creating-microservices/) are JHipster-generated applications (using application type microservice application when you generate them), that handle REST requests. They are stateless, and several instances of them can be launched in parallel to handle heavy loads.
* The [JHipster Console](https://github.com/jhipster/jhipster-console) is a monitoring & alerting console, based on the ELK stack.

In the diagram below, the green components are specific to your application and the blue components provide its underlying infrastructure.



seamlessly