Building a Microservices Application with JHipster 6 and Docker in 30 Minutes

Who am I?

Dennis Sharpe

Chief Technology Officer - Ippon USA

JHipster Fan

Worked with Julien Dubois (JHipster creator & lead developer)

@SharpeDennis

IPPON

Discovery to Delivery

Who are You?

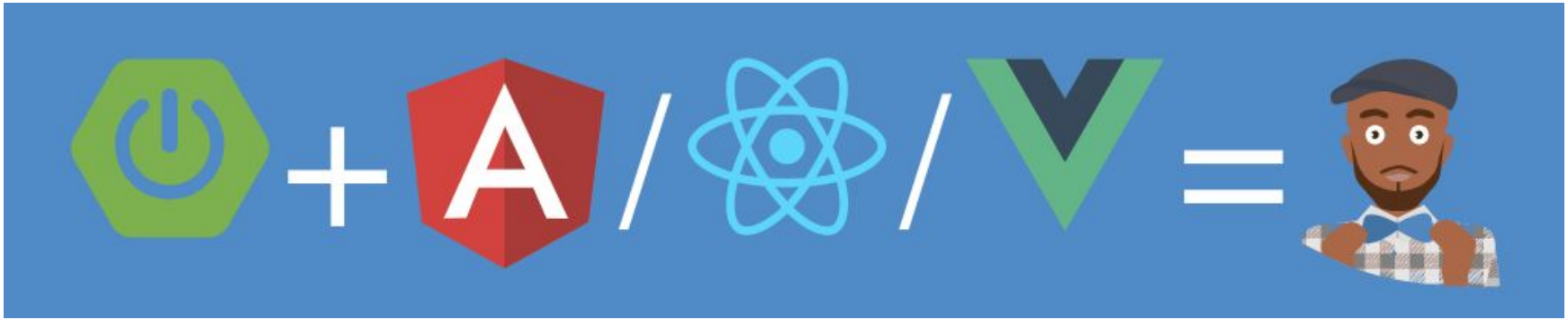
- ★ Don’t start the timer yet!
- ★ First I’ll give an overview

BUT…IF I CAN’T DELIVER IN 30 MINUTES OR LESS…

JHipster is FREE for all attendees!!!

What is JHipster?

- ★ Java web application and microservice generator
- ★ Front-end generated in Angular, React, or Vue.js
- ★ Data model (and entities) generated using command-line or JDL
- ★ CI/CD pipeline generation
- ★ Cloud deployments to AWS, GCP, Heroku, Cloud Foundry, and Azure
- ★ Kubernetes, Docker, or OpenShift deployments
- ★ Complete microservices architecture included (Spring Boot + Netflix OSS)



Frequently Asked Questions

- ★ Is it mean.io?
  - No, it uses Java/Spring on the backend.

- ★ Doesn't the Spring Initializer do that?
  - No, it uses Angular/React/Vue for the front-end.

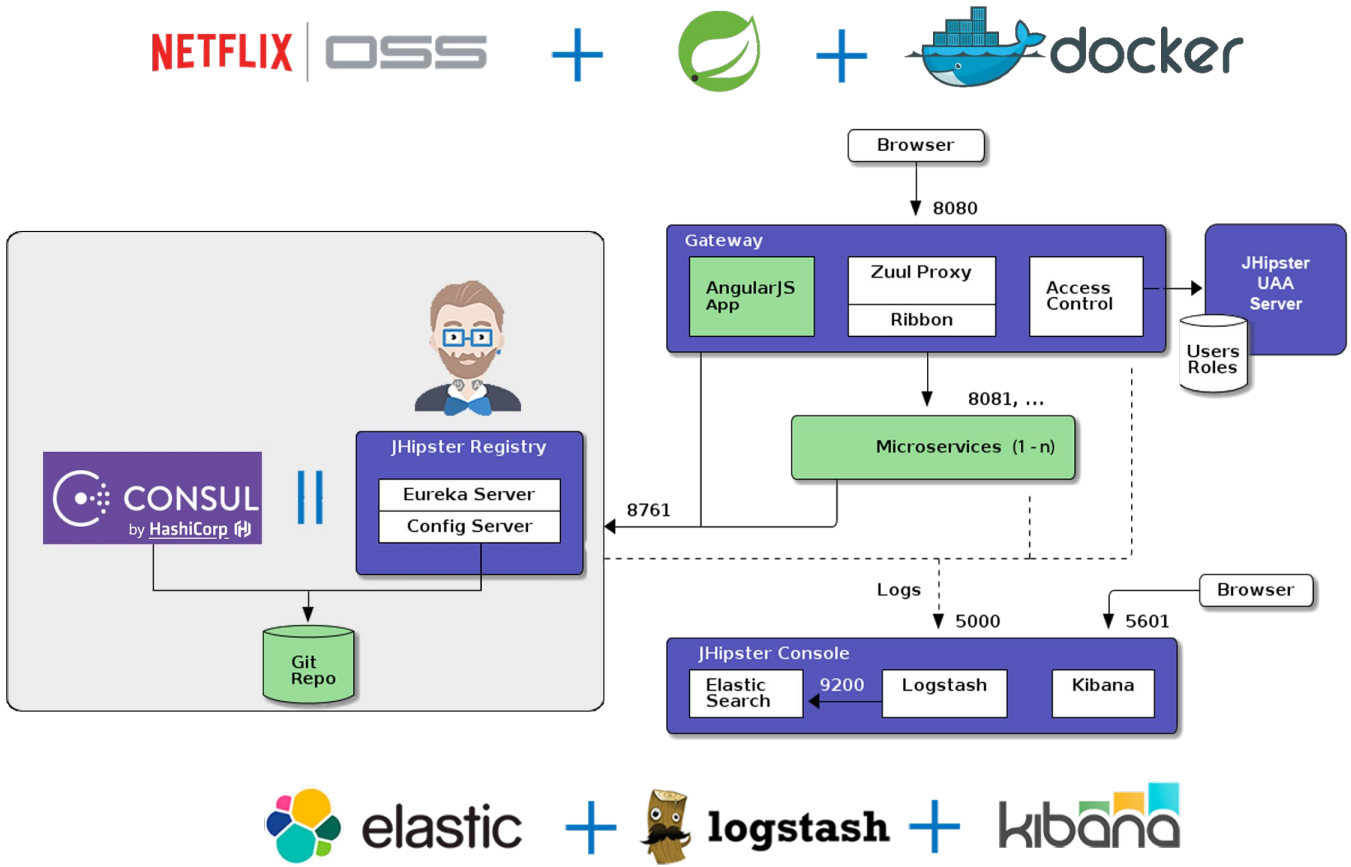
- ★ Why does it not support **LatestFramework.js**?
  - Someone always asks this. Don't be that person. HINT: There is a Marketplace!

Fowler: An approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API

- ★ Scale independently
- ★ Teams and architecture organized around business capabilities (instead of by technical discipline)
- ★ Smart endpoints and dumb pipes
- ★ Teams typically develop, test, deploy and support their microservices

INSERT MICROSERVICES DEBATE HERE -->

The Big Picture



DEMO TIME

```
mvn -Pprod package jib:dockerBuild
```

★ Compiles and tests the production profile (maven and Spring)

- Includes running all the JUnit tests
- Uses Jib (an open-source Java containerizer from Google)

- ★ Microservice
  - No GUI
  - No user management code
- ★ Gateway
  - A “router” to microservices
  - Load balancing and circuit breaking
  - Security and user management
  - Rate limiting
  - Generated UI based on microservices

```
mvn -Pprod verify jib:dockerBuild
```

- ★ Compiles and tests the production profile (maven and Spring)
  - Includes running all the JUnit tests
- ★ Compiles and tests the typescript
  - Includes running all the front-end Jest unit tests

https://jhipster.github.io/api-gateway/

- ★ Routing with Netflix Zuul
- ★ Load balancing with Netflix Ribbon
- ★ Circuit breaker with Netflix Hystrix
- ★ Security using JWT or OAuth2
- ★ Documentation generation with Swagger
- ★ Rate limiting with Bucket4j and Hazelcast

The JHipster Registry

- ★ Runtime component provided by JHipster
- ★ Fully Open Source (Apache 2 license)
- ★ Service Registry based on Spring Cloud Eureka
  - All services register themselves on the JHipster Registry
  - Allows load balancing on the gateways
  - Allows microservice scalability and cluster configuration
- ★ Configuration server based on Spring Cloud Config
  - Sends configuration data to all services
  - Useful to version, tag, rollback configurations
  - Allows you to store “sensitive” information like database passwords

- ★ Monitoring console based on the ELK stack
  - Elasticsearch, Logstash, Kibana
  - Aggregates logs from microservices and gateways
  - Provides pre-defined dashboards
- ★ Logs are sent by each JHipster application
  - Log messages sent by using the logback API: “log.debug()”
  - Dropwizard Metrics data dumped regularly to the logs, with detailed information from the JVM, Spring Beans, etc.
- ★ Alerting is also available
  - Using Elastalert from Yelp

DEMO TIME AGAIN

Source : posts2212.rssing.com/chan-13732161/all_p9.html

- ★ Scale the “catalog” microservice with Docker
  - Run “docker-compose scale catalog=2”
- ★ A second instance of “catalog” is running
  - As it uses HazelCast, a distributed cache will be automatically configured between both instances
  - This second instance will be available in the JHipster Registry and in the gateway’s admin screen
  - It will also be automatically monitored by the JHipster Console
- ★ You can launch new instances, and kill existing ones, to see how the architecture handles failure, circuit breaking and load balancing
- ★ When you have finished, just run “docker-compose down” to destroy

Conclusion

- ★ This stuff is pretty cool
- ★ Even if you can’t use JHipster, use it to learn
- ★ Microservices can be scaled independently and easily with Docker
- ★ Beer is good, let’s have some

Ippon

contact@ipponusa.com

https://blog.ippon.tech/

@IpponTech