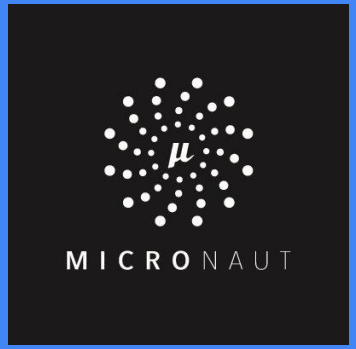
Micronaut in 15 minutes

Paris Apostolopoulos
@javapapo



Disclaimer

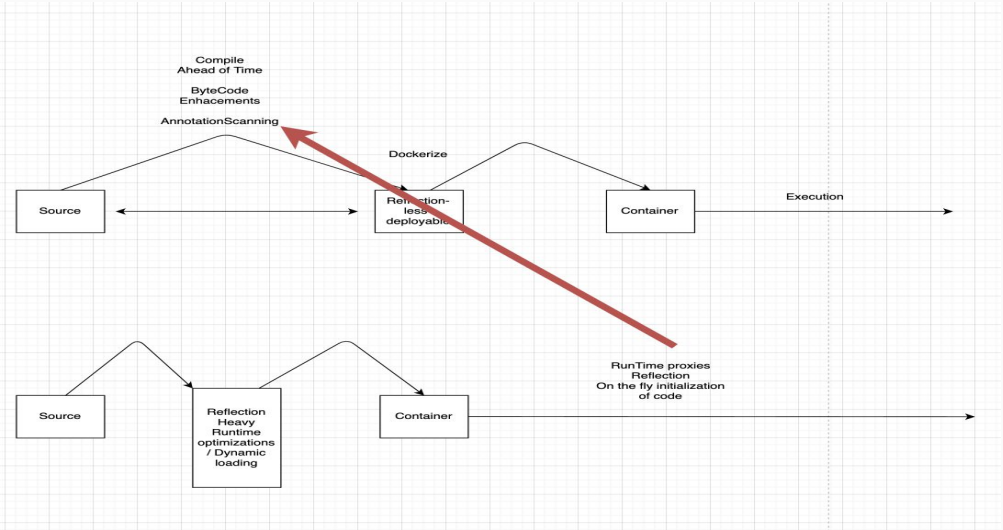
- My past as a Java dev
- Not affiliated with Objectcomputing.com or Oracle
- Opinions are of my own based on real work and attempts to solve real problems (not PoC-es)
- I have been using Micronaut for more than a year full time, on the finance sector.

A new breed of frameworks

- Java moves into the cloud.
  - Already significant progress with frameworks like SpringBoot
- We are still transitioning as an ecosystem
  - Birth of Graal VM
  - Attempts to write Java and execute Native
  - Make Java more cloud friendly, still working on that (java on diet, start up times, resource allocation mem/cpu)
- Micronaut, Quarkus, Helidon, Javalin, sparkjava, ktor.io… SpringBoot evolutions
- The Java ecosystem is sexier than ever!

Shift to the left

- Some of the new comers opt in to fully embrace the concept of Ahead of Time Compilation. The idea is not new.



Birth of Micronaut

- 2018 (https://www.slideshare.net/graemerocher/introduction-to-micronaut-at-oracle-codeone-2018)
- Apache V2 license
- Core contributors from Grails framework, Spring etc
  - Greame Rocher (currently Oracle) @graemerocher
  - Ivan Lopez (ObjectComputing) @ilopmar
  - James Kleeh (ObjectComputing) @Schlogen
  - Sergio Del Amo (ObjectComputing) @sdelamo
  - And others ...but relatively small team.
- Development is sponsored by ObjectComputing.com

Core concepts for transitioning spring boot devs

- Similar development experience with Spring Boot.
  - Dependency management
  - Maven / Gradle
  - Set of familiar annotations
  - Modularized design -similar to spring boot starters, import BOMs.
  - A lot of ready made integrations available (starters)
    - Kafka, ElasticSearch, Caching, DB orms, HttpServers, RxJava big list: https://micronaut.io/documentation.html

But very different at the same time

- Micronaut is heavily based on Dependency Injection and Configuration wiring that happens during compilation (AOT).
- It offers a similar to Spring Boot DI experience but with no use of Reflection or Runtime proxies
- All AOP proxies and config happens during the compile phase.
- The DI mechanism is very similar conceptulaly to implementations like Dagger (https://github.com/google/dagger)

Benefits

- Despite the slightly longer compilation time, you get to have a reflection free core. All enhancements and proxies have been created and wired before you start your application
- Reduced memory footprint *(at least for the framework code)
- Faster start up times, since we dont have to spend cycles to do wiring or init configs

How do I start?

- Head to https://micronaut.io/launch/ (similar to spring.start.io) and create a skeleton app.
- Or use the micronaut CLI
  - `> sdk install micronaut`
  - `> mn create-app test-service`
- Or do it manually with the starter BOMs or use some archetypes :
  https://github.com/gasches/micronaut-archetypes

What about documentation?

- Currently your no.1 documentation destination is the official User Guide: https://docs.micronaut.io/latest/guide/index.html
- It gets updated on every release and it receives regularly contributions by the community.
- There are a lot of specific guides for different integrations see: https://micronaut.io/documentation.html
- There is no known bibliography at the time being.
- Repo with presentations :
  - https://github.com/micronaut-projects/presentations
- Youtube content as well:
  - https://www.youtube.com/results?search_query=micronaut

Community support?

- Over the past year I have seen an increase on questions on StackOverflow and on Gitter around Micronaut. https://stackoverflow.com/questions/tagged/micronaut
- Official Gitter channel (very helpful, members of the core team show up a lot and provide help or tips) . https://gitter.im/micronautfw/ .
    - Do join if you plan to use Micronaut

Any concerns?

- Micronaut is production ready, but at the same time is a framework that is still maturing especially in the `integrations` layer.
- There are not a lot of breaking changes on the core as it has progressed from version 1.x to 2.x
- The community is still expanding, within this past year I think adoption almost doubled - but still you can not beat the Spring Ecosystem
- No bibliography at the moment, maybe subject to change.

Is it worth giving Micronaut a try?

- Definitely yes, it has passed the phase of baby steps and early adoption, it is used in production now.
- But
  - There is definitely a learning curve.
  - Spring friendly devs will find transitioning from Spring Boot to Micronaut easier.
  - Definitely easier when you start from scratch and you don't have to pivot legacy apps.
  - If you already have big codebase with already established services you need to evaluate your current situation. Are you in total control of your code? Can you predict or do you know fully know the weaknesses of your current solution?
    - If yes then trying to migrate is going to be relatively easy

And this means I can do native as well?

- Yes but there are a lot of things to think about
- The fact that micronaut is reflection free does not mean that your custom code, and libs are reflection free as well - know your code.
- The GraalVM ecosystem is still picking up, complex and big SpringBoot applications are not very easy to migrate yet . Things improve
- We are still transitioning, not a lot of experience on running Java Native payloads in production.

Thank you!

Any questions?