QUARKUS

The Bold, the Broken, and the Burned

Hard won lessons
in the 7 years of developing Quarkus

Guillaume Smet, IBM

@gsmet_

&

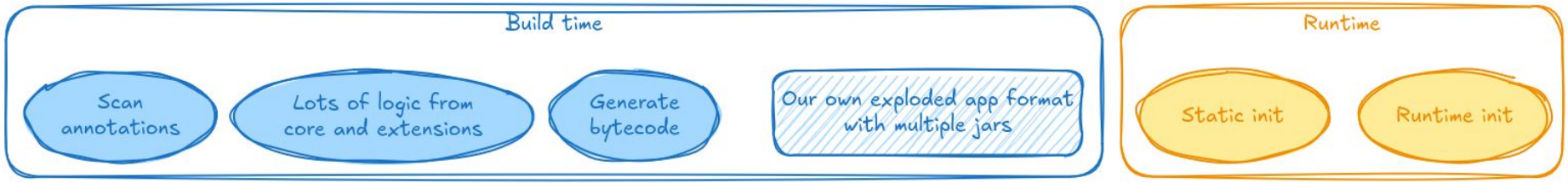
(Georgios Andrianakis, IBM

@geoand86)

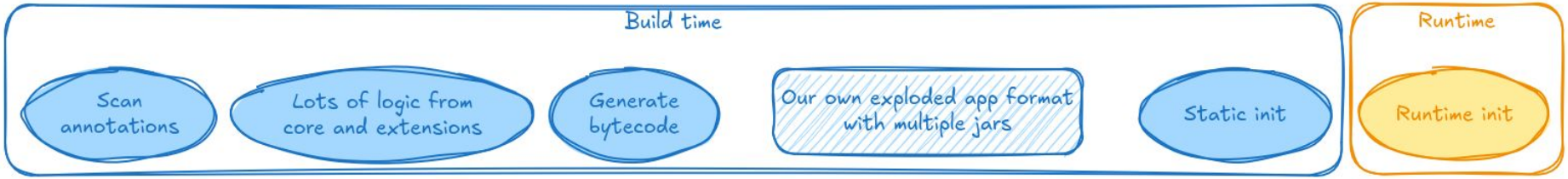
Quarkus is not your typical Java framework

We reinvented how to build a Java app...

JVM



Native executable



...and how to develop a Java app

Dev mode

Live reload

Dev Services

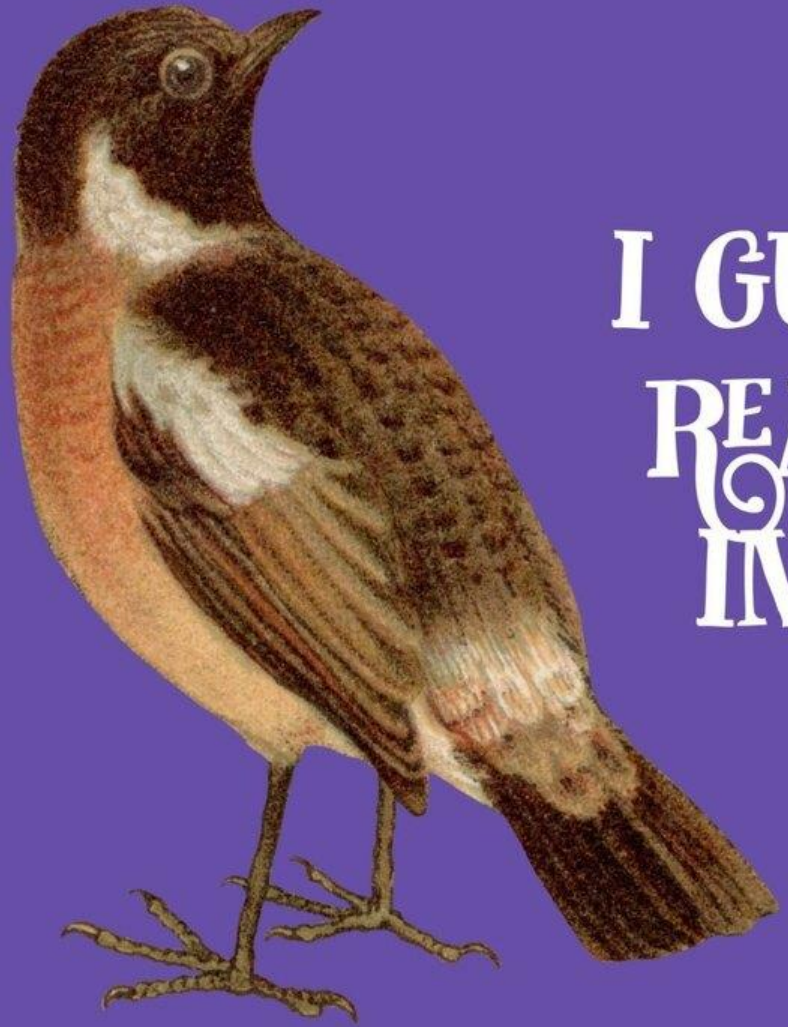
Continuous Testing

It's been 7+ years...
Now is the time for some introspection

What did we do right?
What could have we done better?

We did everything right!

See you next year!



I GUESS YOU DON'T REALLY DO MUCH INTROSPECTION SHIT

@effinbirds

Starting from scratch

Starting from scratch

- Lots of experience from JBoss EAP/WildFly and Swarm/Thorntail
- Starting from scratch is hard, even if we reused a lot of existing components
- Java ecosystem is vast, hard to cover everything... when your architecture is so different
  - Hey! I use this <insert obscure library here> and it doesn't work?
- A lot of infrastructure to set up

Starting from scratch

- Quarkus started as an experiment
- Organic growth since then
- Sometimes hard to find the time to take a step back and regroup

We already fixed some issues
and will address more in Quarkus 4

The basics

Code organization

- Massive monorepo
  - 1300+ Maven modules
  - Easier refactorings and releases
  - Challenging for IDEs (it got better!), for CI, for Central Portal...
  - Harder for occasional contributors

Quarkiverse

- A centralized hub for Quarkus extensions (175+)
  - Empowers the (very active!) community
  - Independent lifecycle
  - Extension template
  - Easy workflow for releasing and documentation
  - Can transfer ownership to new maintainers

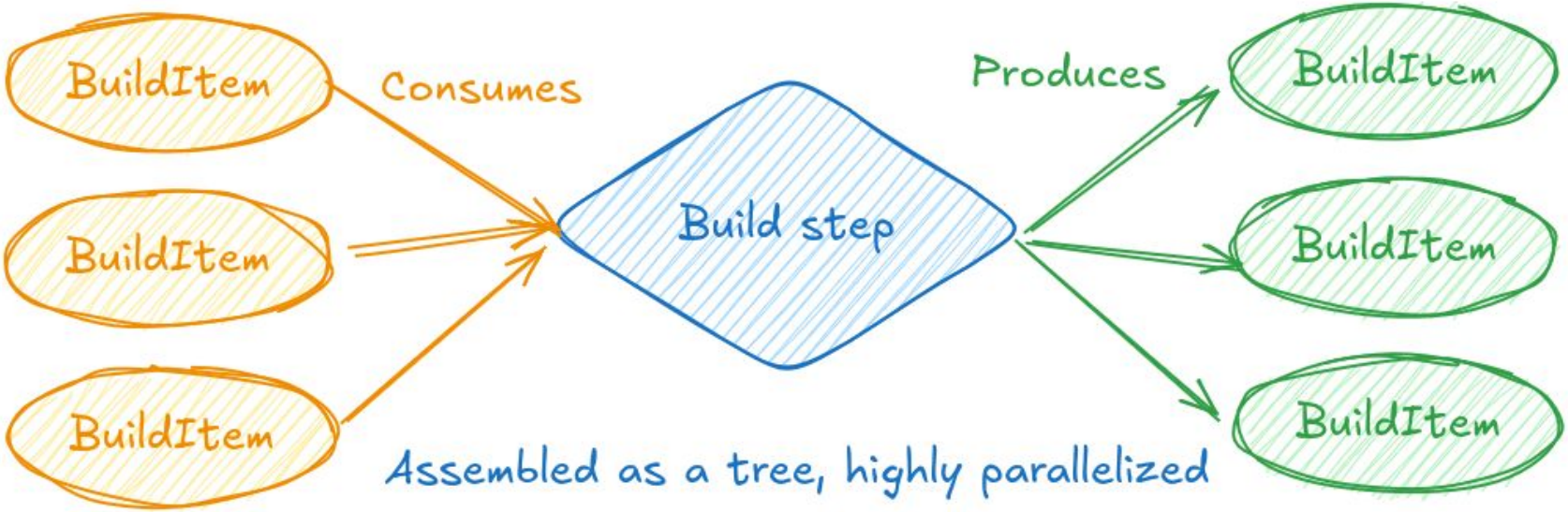
Naming

- "Reactive" everywhere was a huge mistake
- Caused massive confusion
- Done very inconsistently
  - RESTEasy Reactive supported both blocking and reactive workloads
  - Hibernate Reactive is reactive only

We fixed it!

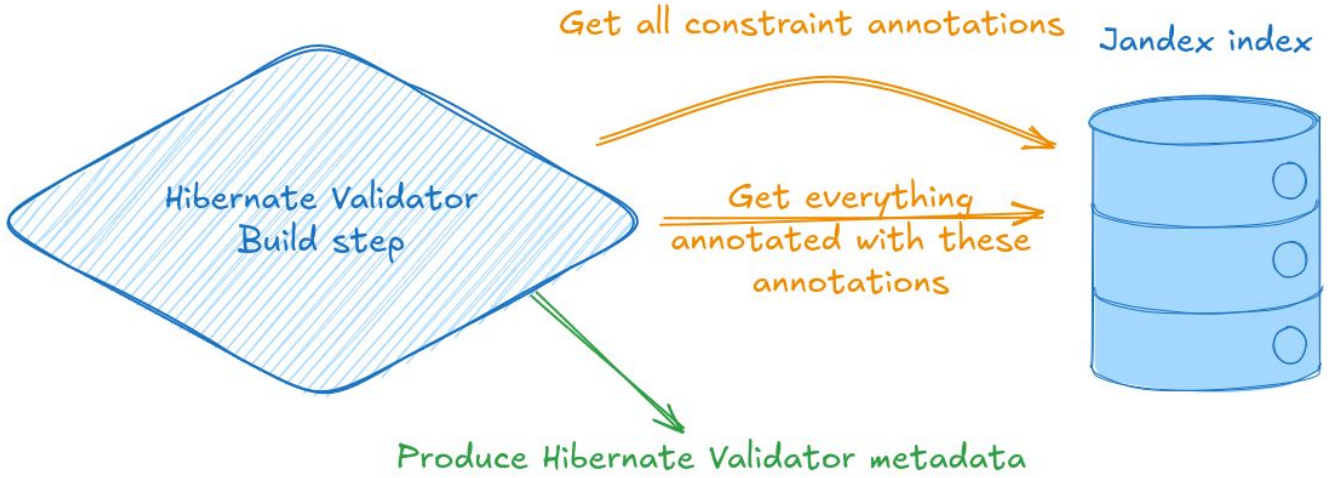
Our build time tooling

Build step framework



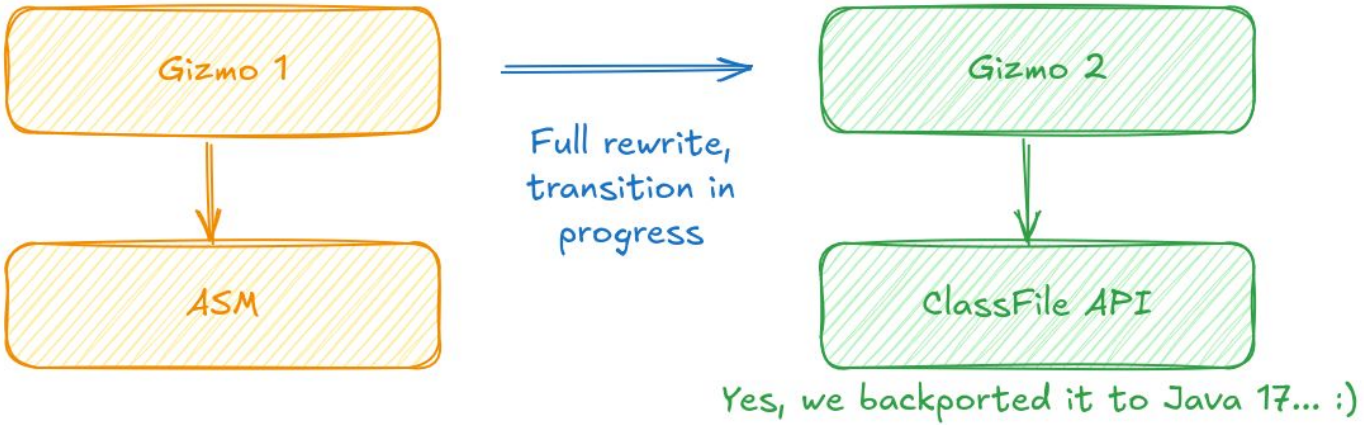
Jandex vs annotation processors

- Index built from classpath
  - Classes, annotations…
- One of the foundations of Quarkus



Gizmo is a blessing

- We generate a LOT of bytecode: that's part of why we are so fast at startup
- Writing everything in ASM: nope!
- Gizmo makes bytecode generation accessible to mere mortals



I'M GETTIN' KIND OF TIRED OF HAVING MY FUCKIN' MIND BLOWN

@effinbirds

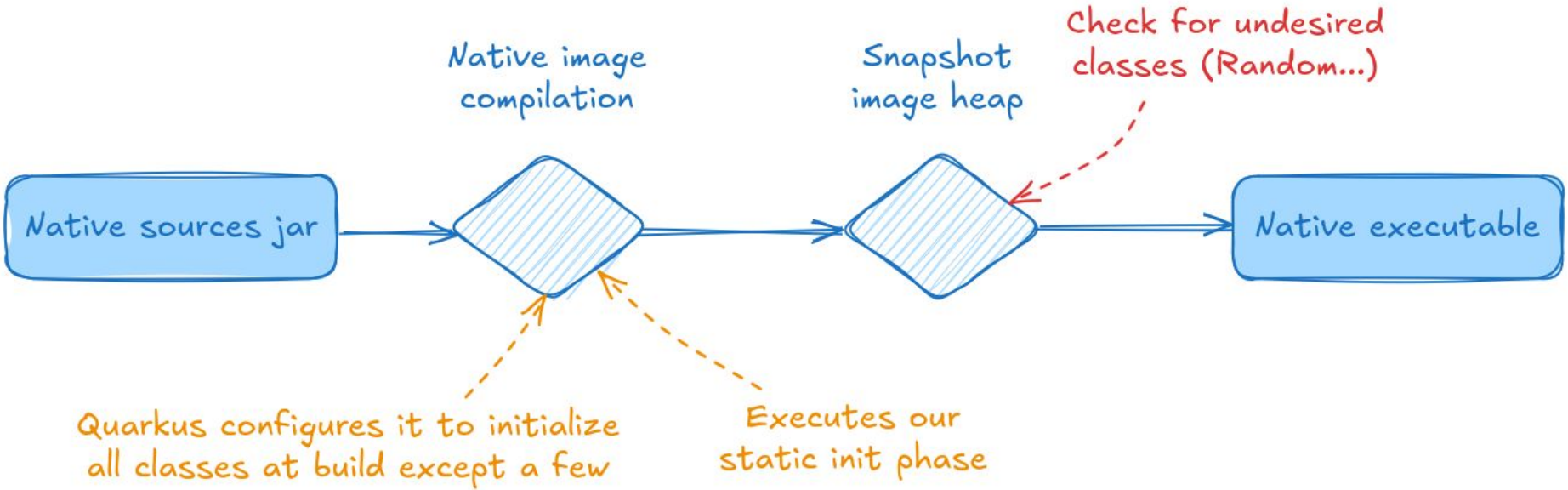
Build tools

- We have our own Maven plugin
  - Build time dependencies not visible at runtime
  - Specific build infrastructure
  - Some work required for Maven 4
- Even harder with Gradle
  - Not much Gradle expertise in the team
  - So many ways to do something: hard to find the right approach

Build reproducibility

- Highly optimized fast-jar layout with multiple jars
- Great but...
- Build reproducibility is harder
  - We need to implement it ourselves
  - Parallelization of build steps + bytecode generation doesn't help!
- We made a lot of progress but still working on it, our target is to get it done by the end of the year

Our vision for native



Our vision for native

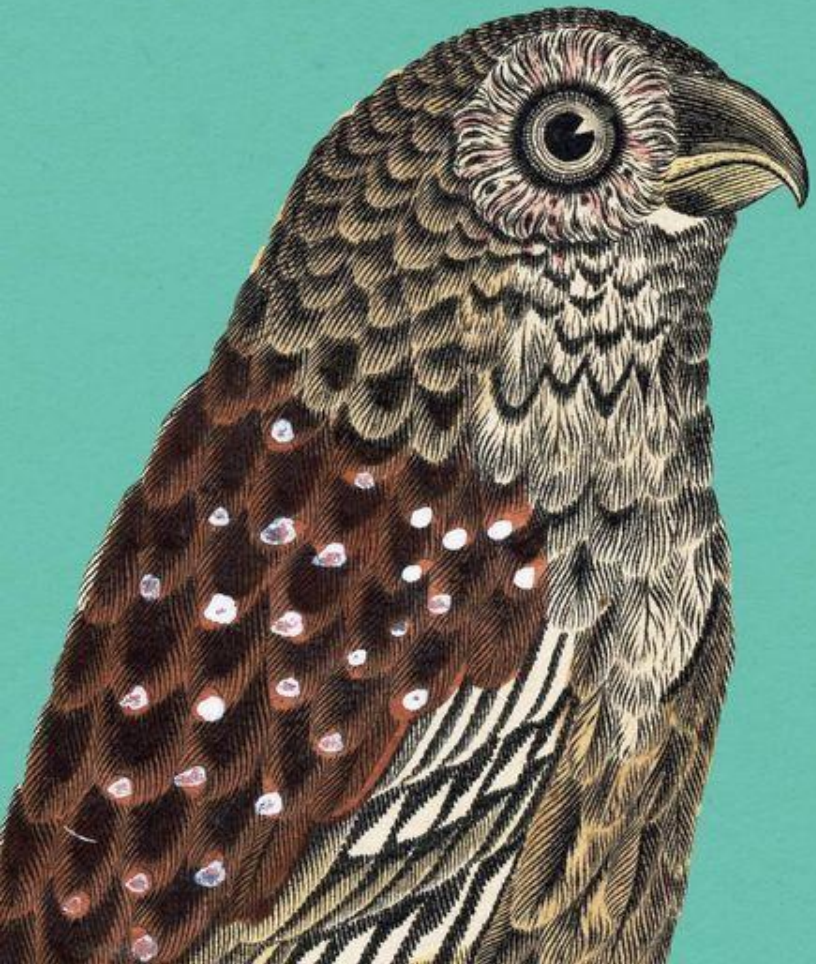
- Maybe not the wisest choice to initialize all classes at build time
  - Causes issues with everything that is not allowed in image heap (Random...)
  - Requires specific native configuration or substitutions too often for external libraries -> requires extensions

We are reconsidering this choice

Dev mode and test infrastructure

Dev mode and test infra

- Truly innovative
  - Live reload actually working
  - Continuous testing
  - Led to lots of awesome features (Dev Services, Dev UI...)
- But...

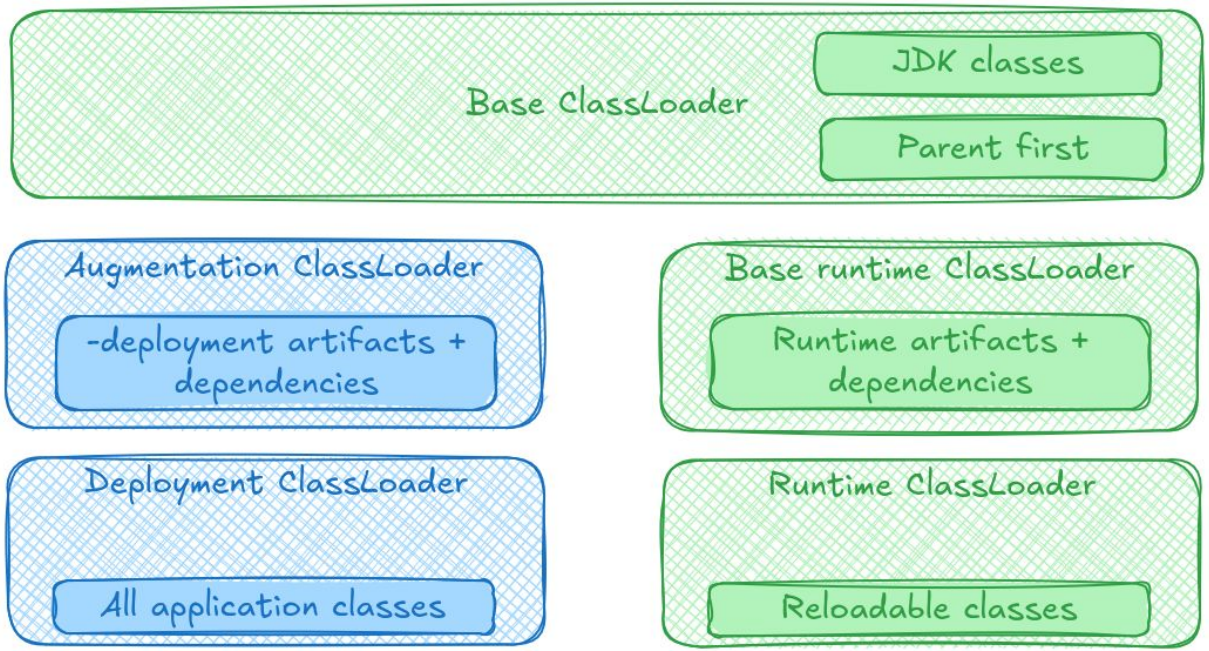


I knew it was going to be a clusterfuck, but this is some huge-ass clusterfuck

@effinbirds

Complex class loading

- *Can be challenging with external libraries*



API choices

Vert.x + Netty for all I/O

- Battle-tested I/O layer
- Very quick reactions to CVEs
- Very small footprint
- Future proof: HTTP/3, io_uring
- Not the easiest thing to work with for maintainers

Vert.x is a large investment that
definitely keeps on paying off

The big dilemma

- Jakarta/MicroProfile standards vs...
- De facto standards vs...
- Our own APIs

Hard to find the right balance...

Panache data layer

- A good example...
- Too innovative/foreign for Java devs?
- Spring Data JPA is widely used...
- Spring JDBC vs Hibernate StatelessSession

Quarkus Data now available with Jakarta Data support

Observability

- Started with MicroProfile Metrics -> moved to Micrometer
- Started with OpenTracing -> Moved to OpenTelemetry
- Logging...
- Cost of migrating + cost of maintaining both for a while...

What's next?

JSON

- Started with JSON-B as default because MicroProfile
- Moved to Jackson as default
- Still supporting both
  - Choice for users
  - Some of our MicroProfile extensions require JSON-B

Jackson 3 anyone?

Community

We got lucky!

- Not only luck, working hard every day to create a nice contribution experience
- A lot of very enthusiastic early contributors
- Some still around but life happens
- New ones popping up regularly
- Hard for occasional contributors to keep up with our fast pace

Big kudos to our 1200+ contributors!

We got help!

- GitHub gave us some additional runners
- Gitflow-incremental-builder by Falko Modler and now Scalpel by Guillaume Nodet
- Tamas Cservenak on everything Maven
- Gradle Inc. donated a Develocity instance and helped us introduce caching into our Maven build
- RunsOn donated a free key - we offload part of our CI to AWS instances
- Lots of interactions with Moderne about OpenRewrite

Thanks!

Being open

Commonhaus Foundation

commonhaus.org

On being open

- Opening the code is the easy part...
- How about:
  - Meetings -> community calls every other week
  - Design discussions -> working groups
  - Decisions -> ADR
- And how to make it sustainable for an occasional contributor?

Releasing (very) often

Releasing (very) often

- Crucial at the beginning, we iterated fast
- Still releasing fast:
  - New minor every month
  - New maintenance micro every week
  - + Emergency releases for CVEs
- Still appreciated but some people got tired of updating

LTS release every 6 months
Maintained for 12 months, new micros every other month

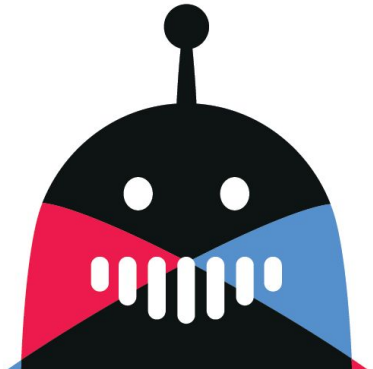
Automation

JUST LET ME DO MY JOB FOR FUCK'S SAKE

@effinbirds

Automation FTW

- Incremental steps towards almost full automation
- Quarkus GitHub Bot
  - Handles all sorts of tasks in our repositories
- More detailed build reporting
- Conversational Release Action
  - Pilot our releases from a GitHub issue
- All in Java backed by Quarkus GitHub App and Quarkus GitHub Action



Ecosystem CI

- A centralized vision of the health of our ecosystem

Quarkiverse	❗
[CI] - Amazon Alexa + Quarkus main	✅
[CI] - Amazon Services + Quarkus main	✅
[CI] - Antivirus + Quarkus main	✅
[CI] - arangodb-client + Quarkus main	✅
[CI] - Artemis + Quarkus main	✅
[CI] - AsyncAPI + Quarkus main	✅
[CI] - Azure Services + Quarkus main	✅
[CI] - Barcode + Quarkus main	✅
[CI] - Batik + Quarkus main	✅
[CI] - Chappie + Quarkus main	✅
[CI] - Config Extensions + Quarkus main	✅
[CI] - Cucumber + Quarkus main	✅
[CI] - Dashbuilder + Quarkus main	✅

Why our own stuff?

- Supply chain attacks anyone?
- No foreign GitHub Actions
- Very cautious about external GitHub Apps

\+ More dogfooding makes a lot of sense

Marketing

I READ SOME SHIT ON THE INTERNET, I'M AN EXPERT NOW

Marketing is hard!

- Too much emphasis on Reactive?
- Too much emphasis on native?
- Too tied to cloud native or microservices?

We are different but you can build any app with Quarkus!

This message wasn't clear enough.

AI

LangChain4j

- Embraced 3 years ago
  - Red Hat / IBM is driving most of the innovation
- Most popular Quarkiverse extension

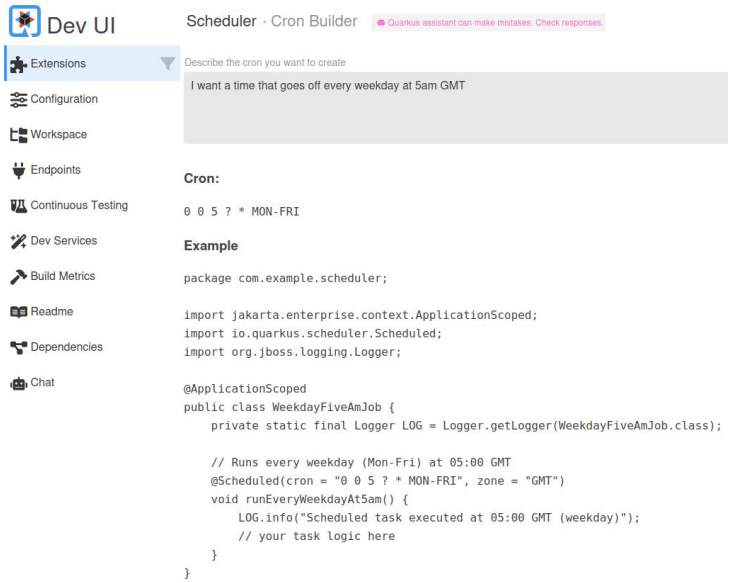
It pays off to be an early adopter
and put in the hard work

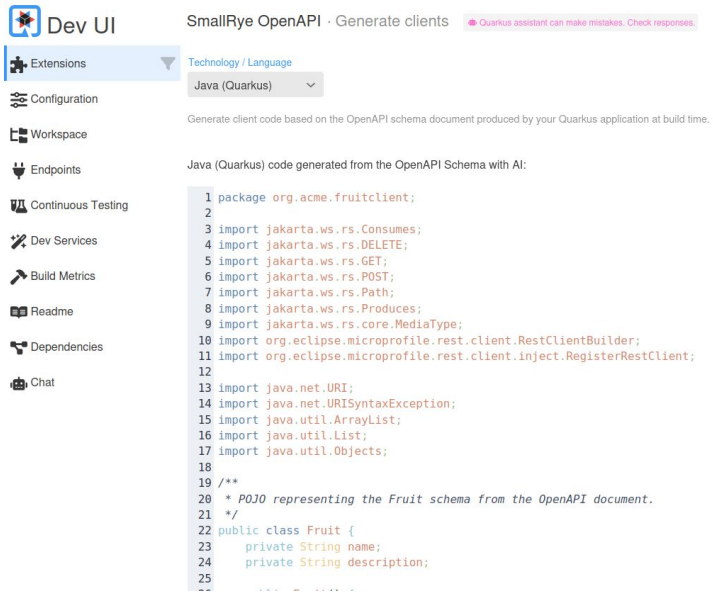
Model Context Protocol

- *Supports both client and server*
- *Supports all transports*
- *Baked-in security*

Having built up a lot of technical expertise over the years has allowed us to adopt MCP at breakneck speed

AI infused development





Separate dev execution mode is super powerful!

Conclusion

Conclusion

- Good ideas
- Some less than great choices
- Lots of lessons learned and being applied!

Thanks!

Illustrations courtesy of:

- Aaron Reynolds (@effinbirds)
- Reza Farazmand (Poorly Drawn Lines)


I'M LEAVING.
WHY?
BECAUSE PEOPLE KEEP ASKING ME QUESTIONS.
POORLY DRAWN LINES