

OpenShift Developer

Architecture Workshop

Introduction to OpenShift for Developers



[linkedin.com/company/red-hat](https://www.linkedin.com/company/red-hat)



[facebook.com/redhatinc](https://www.facebook.com/redhatinc)



[youtube.com/user/RedHatVideos](https://www.youtube.com/user/RedHatVideos)



twitter.com/RedHat

Self introduction

Name: Wanja Pernath

Email: wpernath@redhat.com

Base: Germany (very close to the Alps)

Role: EMEA Technical Partner Development Manager

- OpenShift and MW

Experience: Years of Consulting, Training, PreSales at
Red Hat and before

Twitter: <https://twitter.com/wpernath>

LinkedIn: <https://www.linkedin.com/in/wanjapernath/>

GitHub: <https://github.com/wpernath>



First book just published

Getting GitOps

A technical blueprint for developing with Kubernetes and OpenShift based on a REST microservice example written with Quarkus

Technologies discussed:

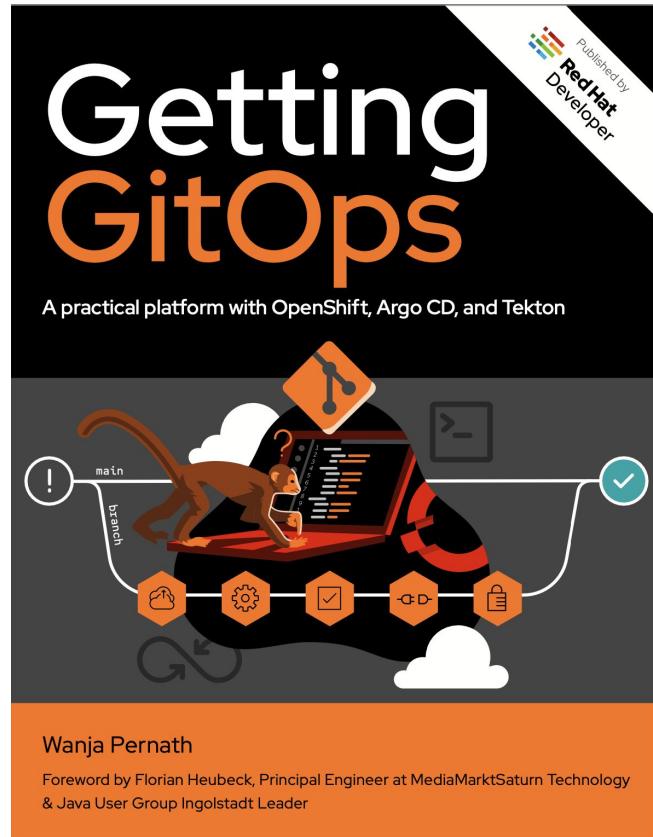
Quarkus, Helm Charts, Kustomize, Tekton Pipelines, Kubernetes Operators, OpenShift Templates, ArgoCD, CI/CD, GitOps....

Download for free at:

<https://developers.redhat.com/e-books/getting-gitops-practical-platform-openshift-argo-cd-and-tekton>

Interview with full GitOps Demo:

https://www.youtube.com/watch?v=znMfVqAIRzY&ab_channel=OpenShift



Agenda

Agenda

- What is this all about
- The OpenShift Ecosystem for Developers
 - Basics
 - Advanced topics
- Quarkus Business Value
- Quarkus Technical Value

What is this all about?

Learning Goals

- Getting an overview of the whole OpenShift ecosystem for developers
- Learn about the benefits of OpenShift prior plain Kubernetes
- Learn how to effectively and efficiently demo OpenShift for developers
- Learn to use OpenShift and its ecosystem
- Learn to start coding with OpenShift in your preferred language
- Learn to do proper release management with OpenShift; understand the basics and how to include it into your release management process but also learn how to benefit from OpenShift Pipelines and when to use what
- Understanding and using Operators for releases
- Understand and make use of advanced OpenShift features like Istio and Serverless

The Use Case

Source Code for this workshop:

(Fork it, clone it, use it. It's open source!)

<https://github.com/wpernath/quarkus-grumpycat>

The Use Case

- In order to understand Quarkus and how great it integrates into all the necessary technologies, let's write a game. Let's call it **Quarkus GrumpyCat**
- It's a multi player game for up to 4 concurrent players
- Goal is to be last dog standing



The Use Case

The server is stateless
and maintains the
communication
between the 4 players
in a game



The Use Case

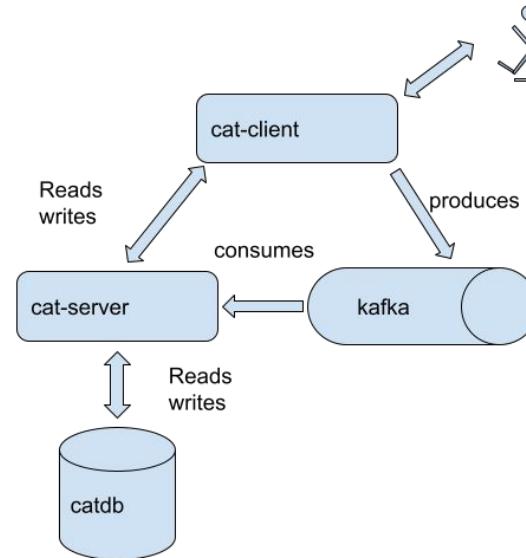


The Use Case

- The client is fully written in JavaScript with a simple game framework called MelonJS (github.com/melonJS/melonJS)
- The client runs in any recent browser (Chrome, Safari, Firefox have been tested)
- The client communicates with the server via
 - REST API and
 - WebSocket
- But, this is a Quarkus workshop (or: a Kubernetes Native Workshop, so we try to ignore the client here, but trust me: Writing a JavaScript game is - pheww).

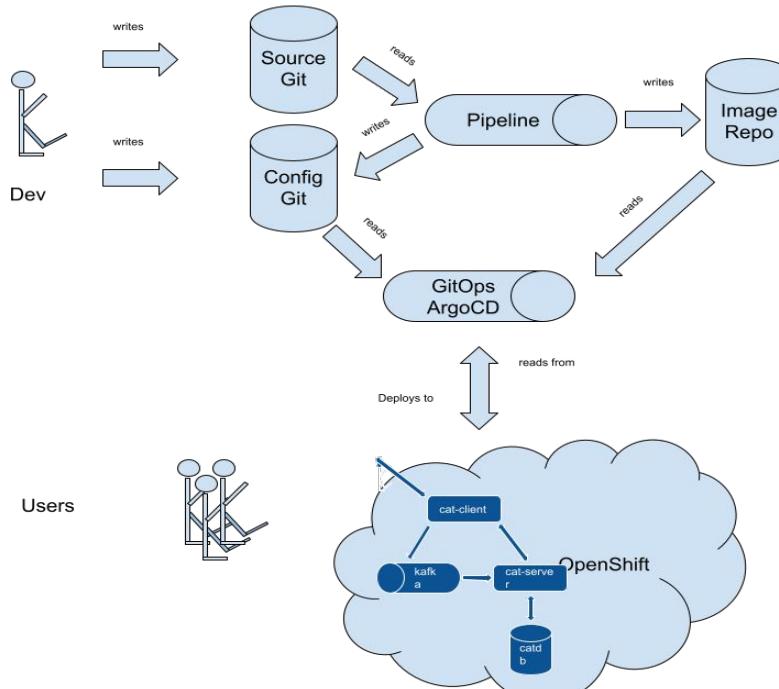
Architecture of the Game

- The user talks via Browser to the cat-client.
- The client talks to the cat-server
- The server has integrations for the database to store state and high scores
- The server also talks to Kafka

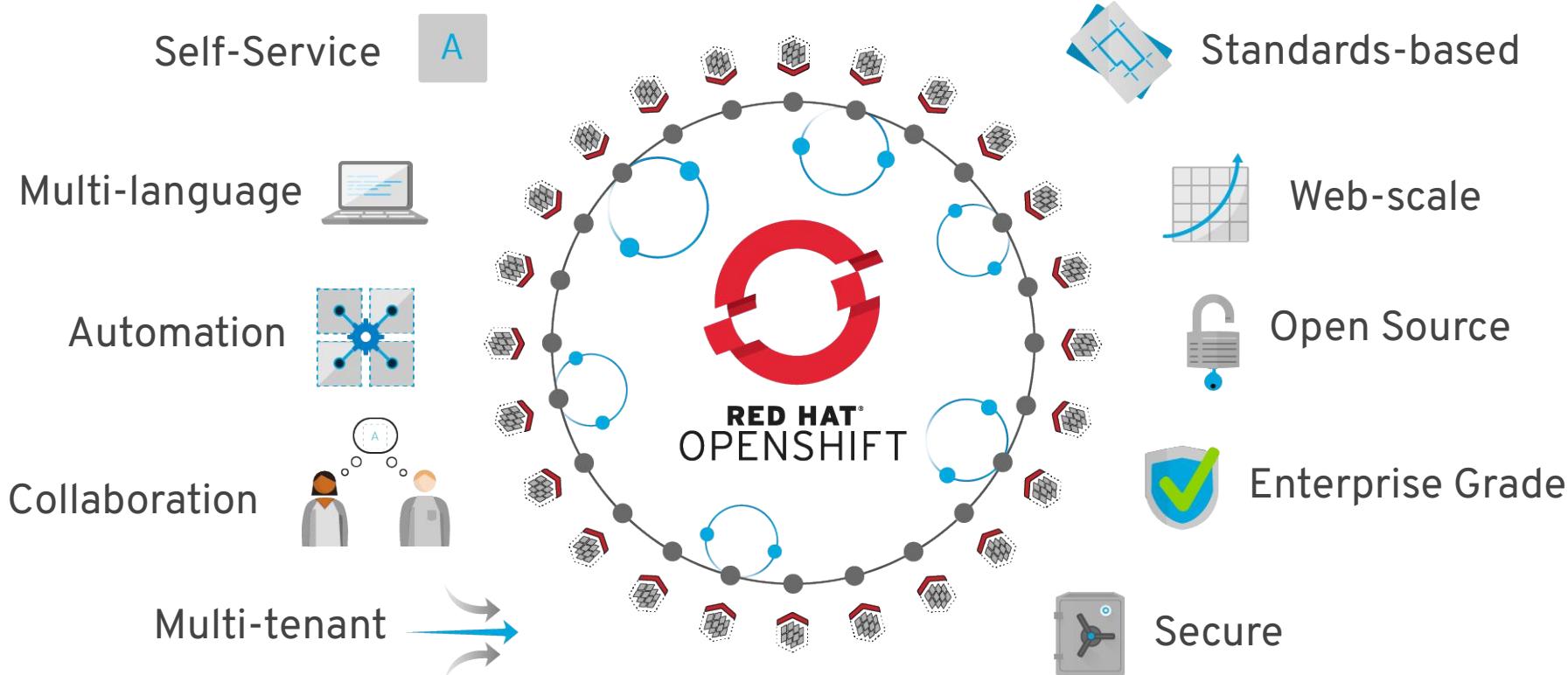


Getting GitOps

- We use the GitOps paradigm for this project, so we will also implement Tekton Pipelines and will use ArgoCD on OpenShift to do CI/CD
- At the end of this workshop you're going to have a blueprint for your own projects.



OpenShift Developers Basics



Value of OpenShift

Monitoring, Logging,
Registry, Router, Telemetry

Cluster Services

Service Mesh, Serverless,
Middleware/Runtimes, ISVs

Application Services

Dev Tools, CI/CD,
Automated Builds, IDE

Developer Services

Automated Operations

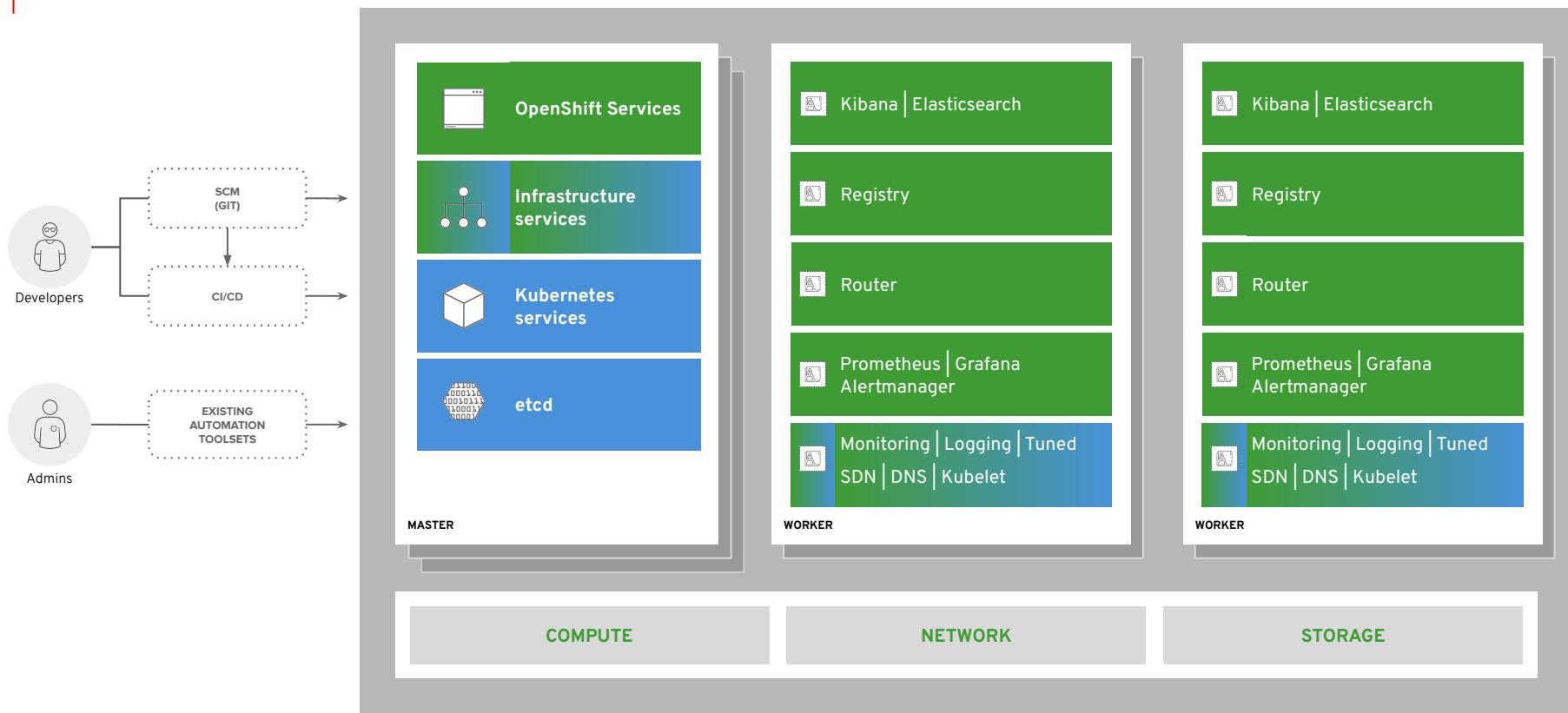
Kubernetes

Red Hat Enterprise Linux | RHEL CoreOS

Best IT Ops Experience

CaaS \longleftrightarrow PaaS \longleftrightarrow FaaS

Best Developer Experience



Overwhelmed? Please see the CNCF Trail Map. That and the interactive landscape are at l.cncf.io

Overwhelmed? Please see the CNCF Trail Map. That and the interactive landscape are at l.cncf.io											
App Definition and Development			Streaming & Messaging			Application Definition & Image Build			Continuous Integration & Delivery		
Orchestration & Management			Scheduling & Orchestration			Coordination & Service Discovery			Remote Procedure Call		
Runtime			Cloud-Native Storage			Container Runtime			Cloud-Native Network		
Provisioning			Automation & Configuration			Container Registry			Security & Compliance		
Cloud			Public			Special			Kubernetes Certified Service Provider		
 <i>This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.</i>											
l.cncf.io											
Platform											
Observability and Analysis											
Logging											
Tracing											
Chaos Engineering											
Serverless											
Kubernetes Training Partner											

Core Concepts

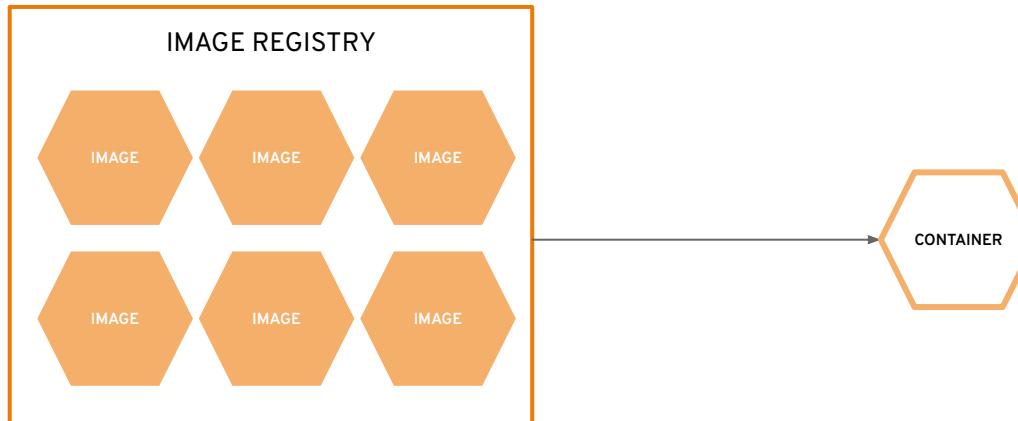
a container is the smallest compute unit



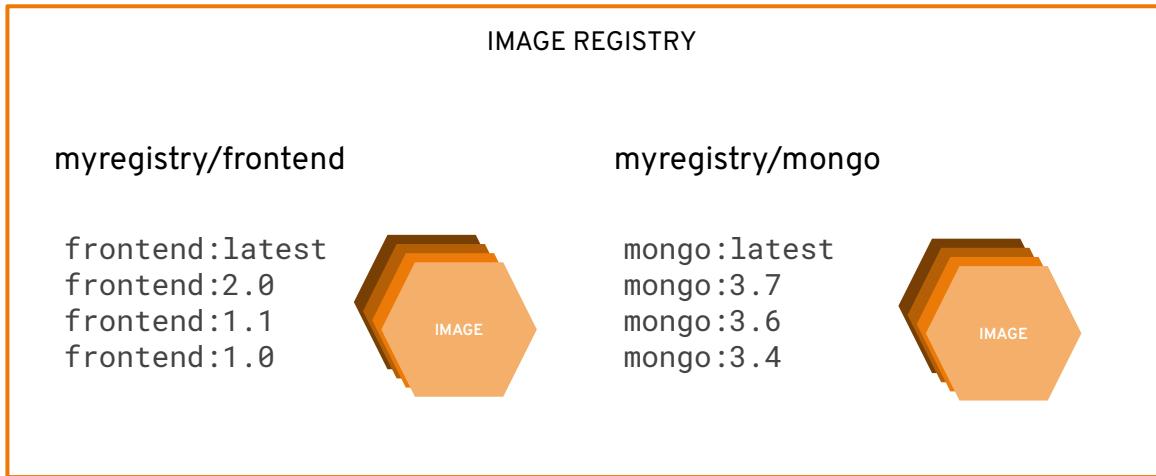
containers are created from container images



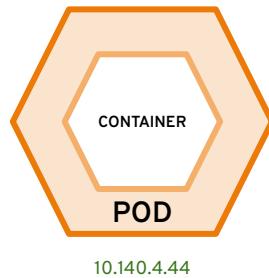
container images are stored in an image registry



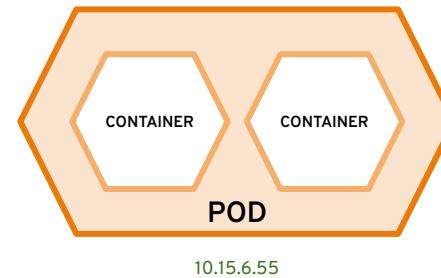
an image repository contains all versions of an image in the image registry



containers are wrapped in pods which are units of deployment and management

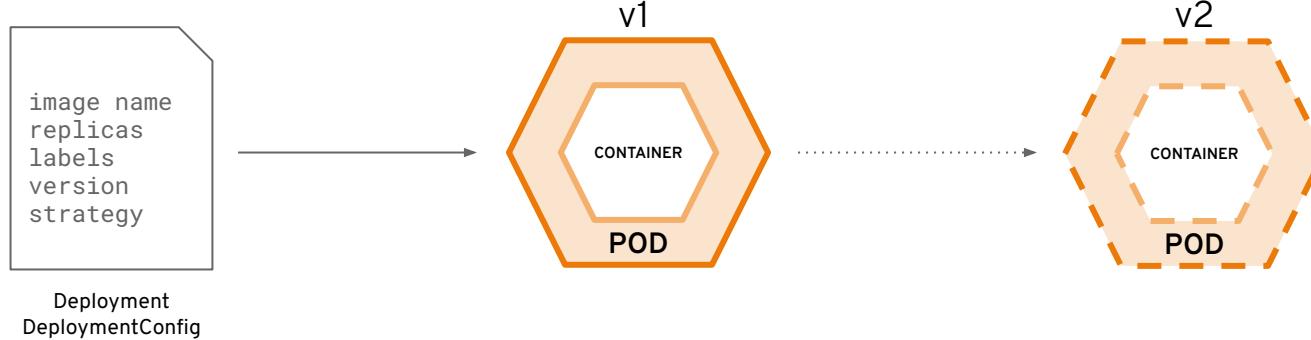


10.140.4.44

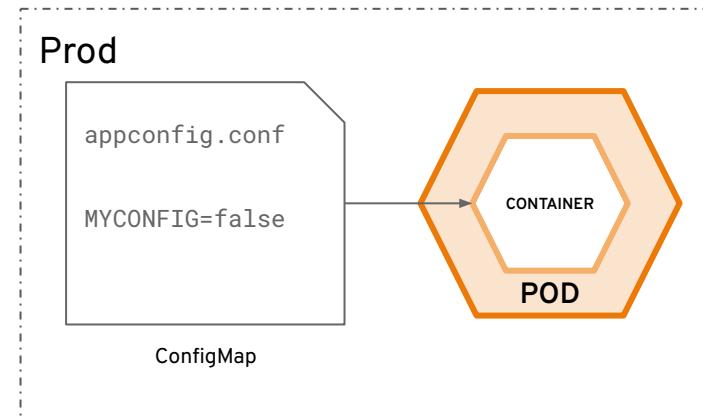
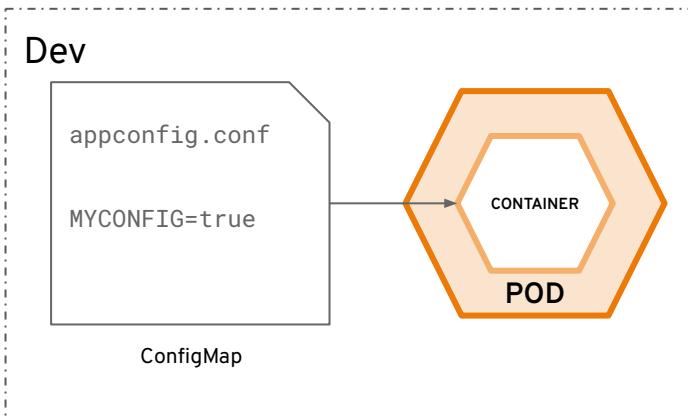


10.15.6.55

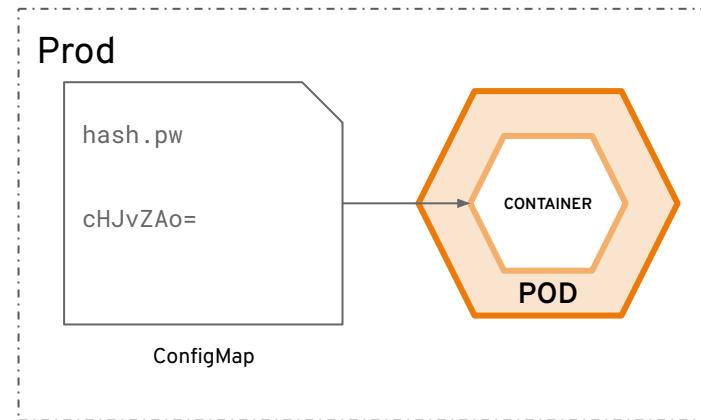
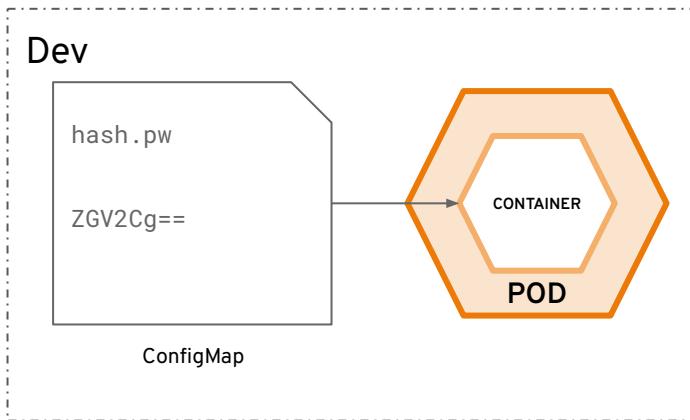
Deployments and DeploymentConfigurations define how to roll out new versions of Pods



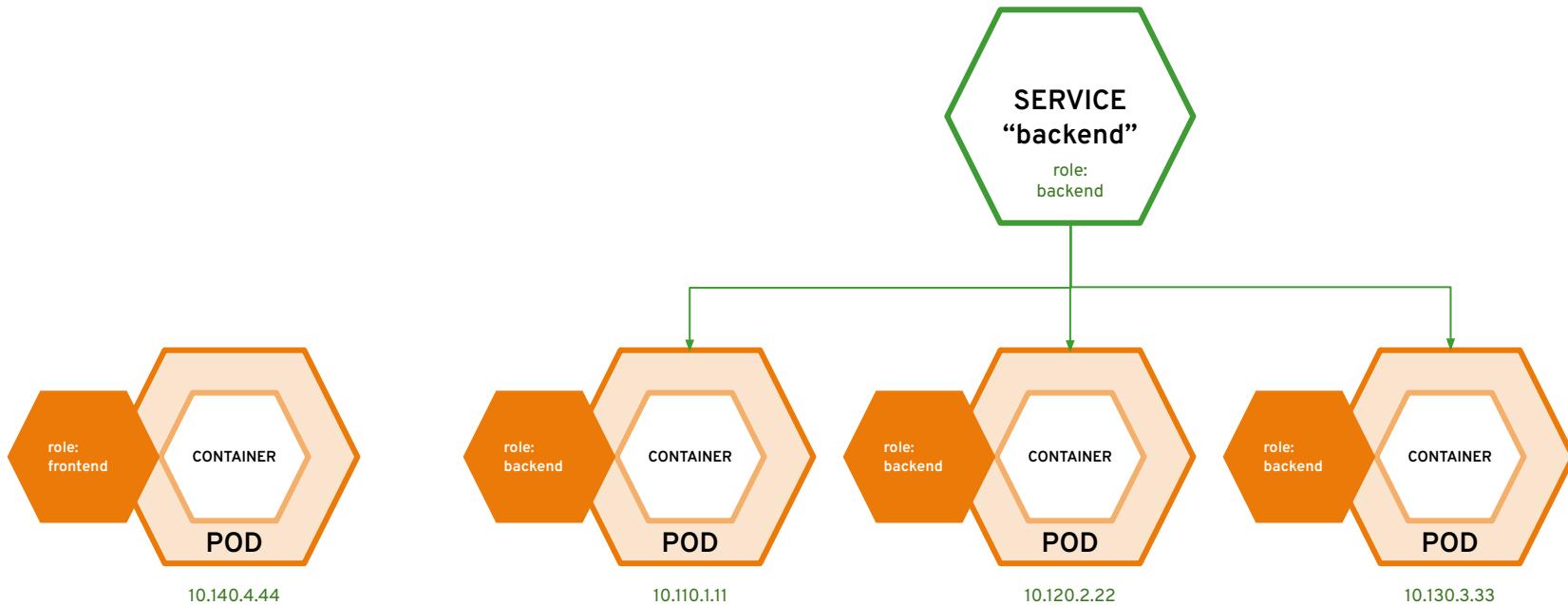
configmaps allow you to decouple configuration artifacts from image content



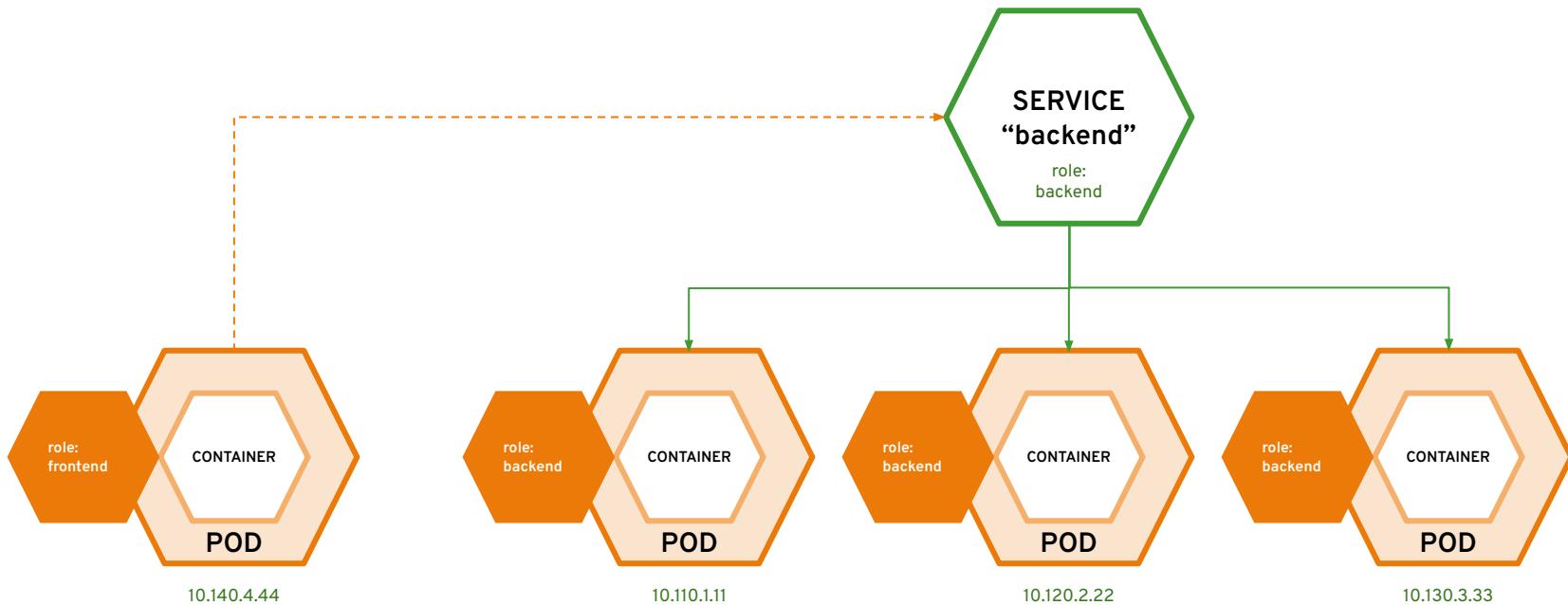
secrets provide a mechanism to hold sensitive information such as passwords



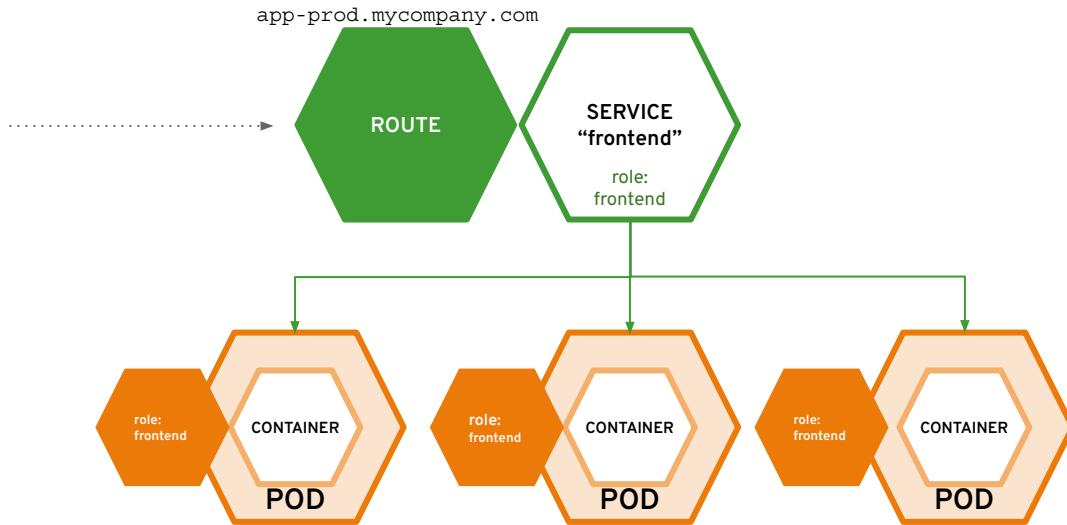
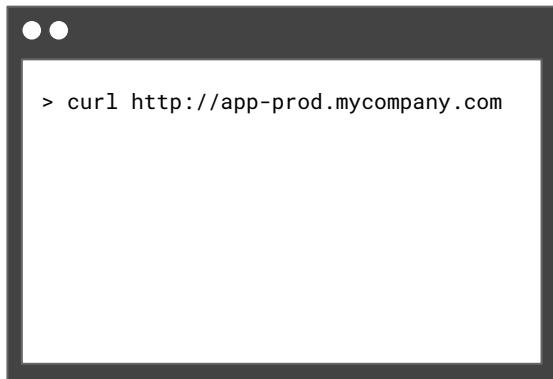
services provide internal load-balancing and service discovery across pods



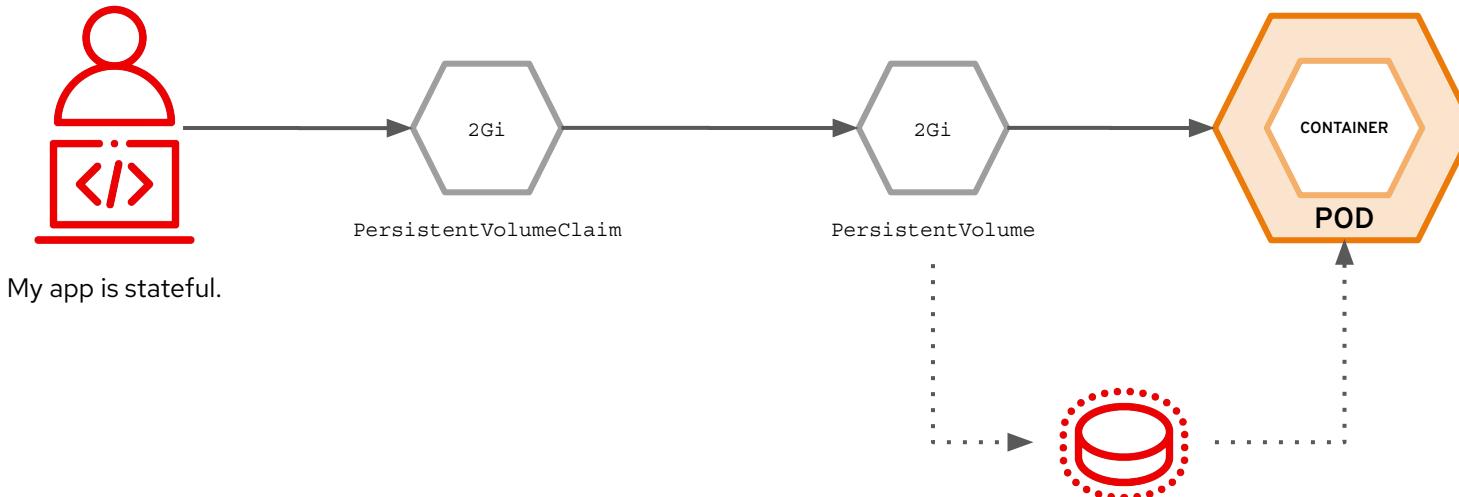
apps can talk to each other via services



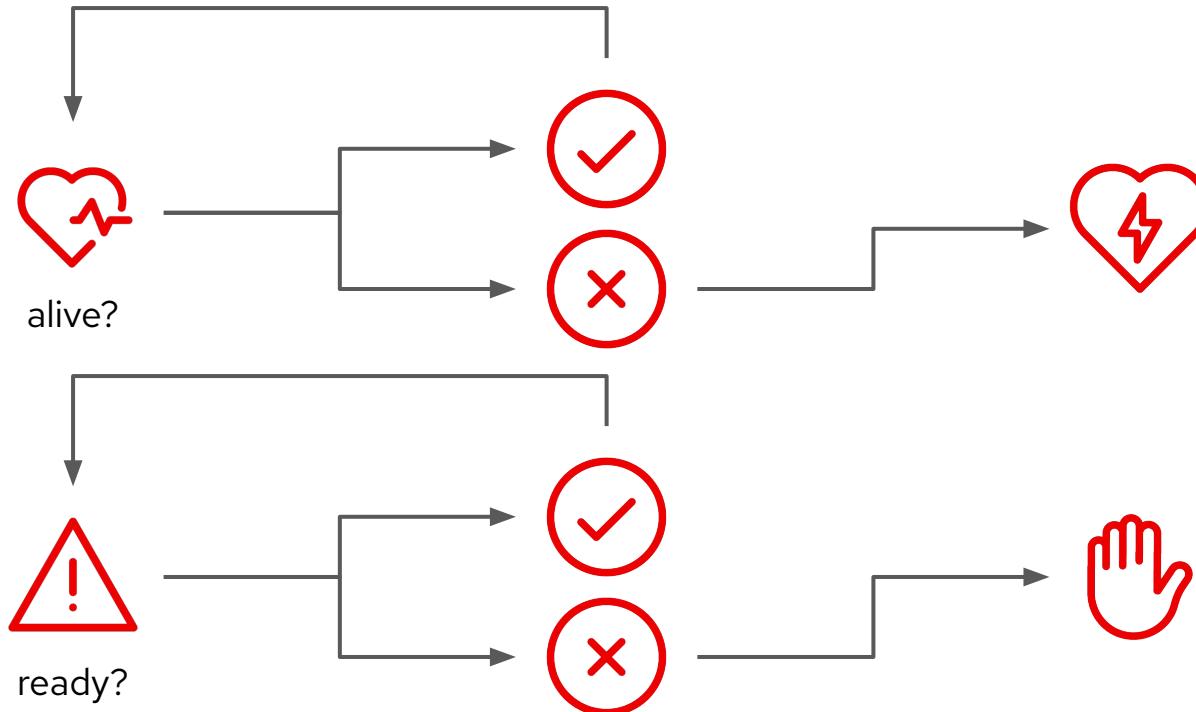
routes make services accessible to clients outside the environment via real-world urls



Persistent Volume and Claims



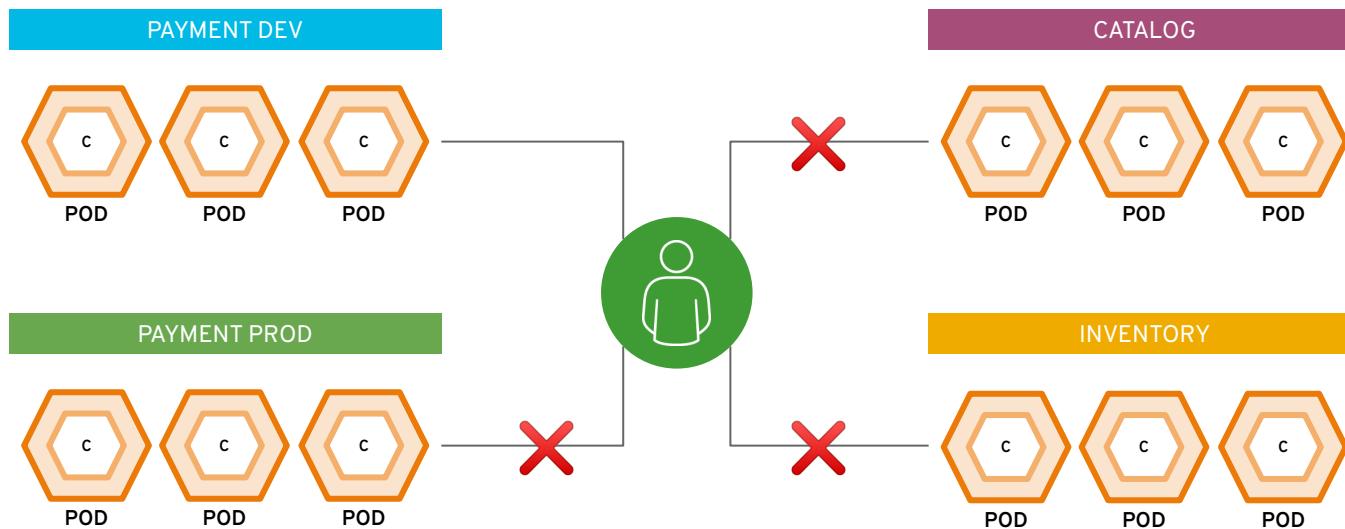
Liveness and Readiness



A note on Probes and Limits / Requests

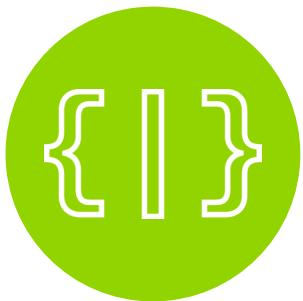
- Your services always MUST have probes configured in the Deployment
 - Kubernetes Scheduler needs to check if your service is up/running
 - Otherwise the default “ping” would be used, which is not necessarily what you want (the target pod could still be reachable, but your service is out of memory etc.)
- You always HAVE to provide resource requests (memory / cpus) to your service' Deployment and it MUST NOT be the minimum required!
 - The Kubernetes Scheduler needs to know WHERE to place your service
 - Resource request should be set to numbers your service typically runs with
 - In Java: `-Xms = -Xmx <= request`

projects isolate apps across environments,
teams, groups and departments



BUILD & DEPLOY

BUILD AND DEPLOY CONTAINER IMAGES



DEPLOY YOUR
SOURCE CODE

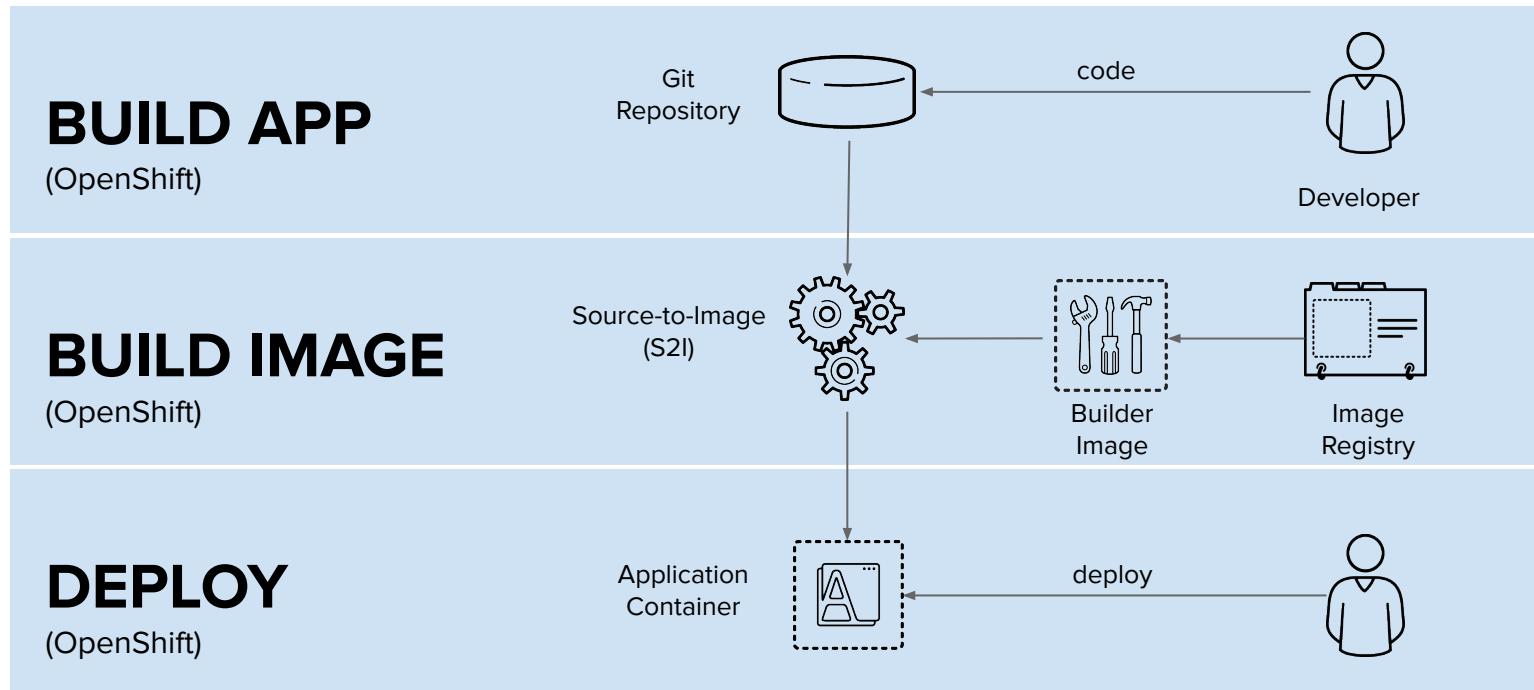


DEPLOY YOUR
APP BINARY

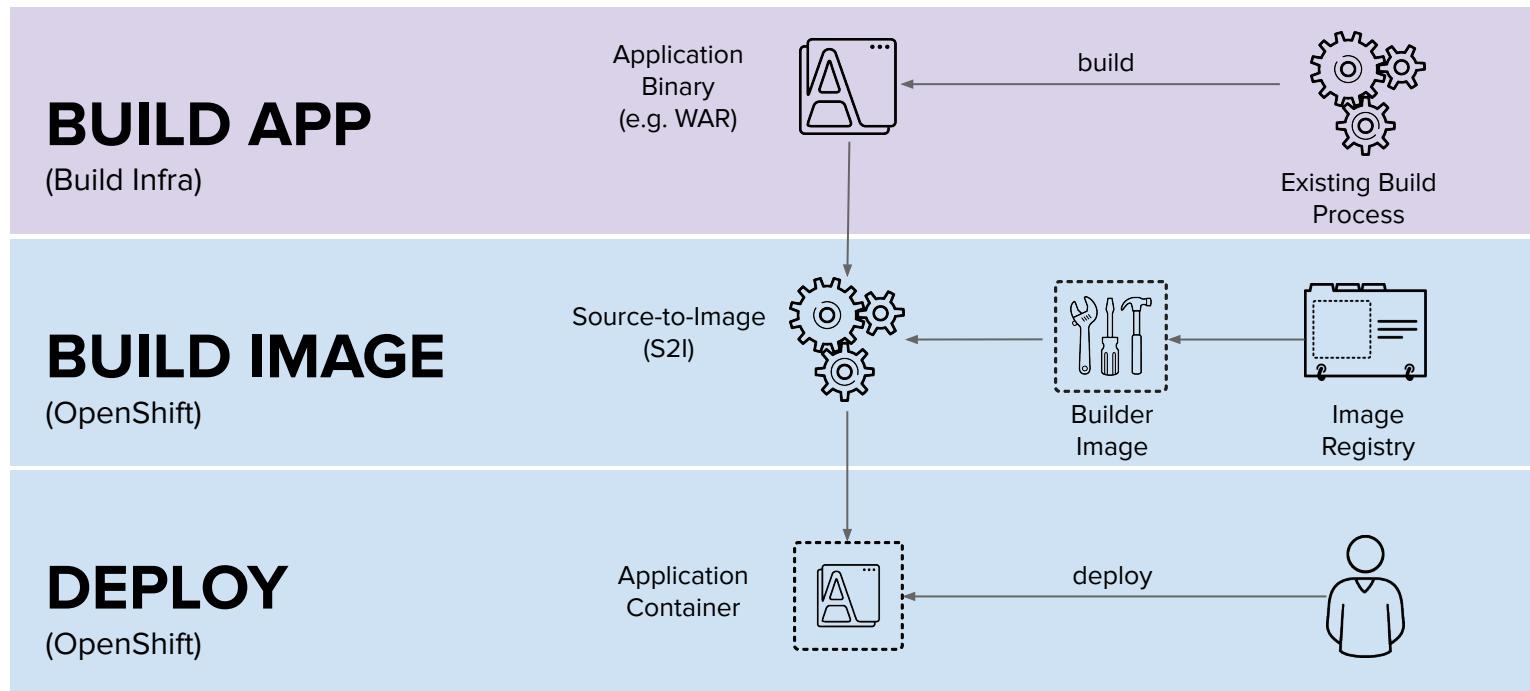


DEPLOY YOUR
CONTAINER IMAGE

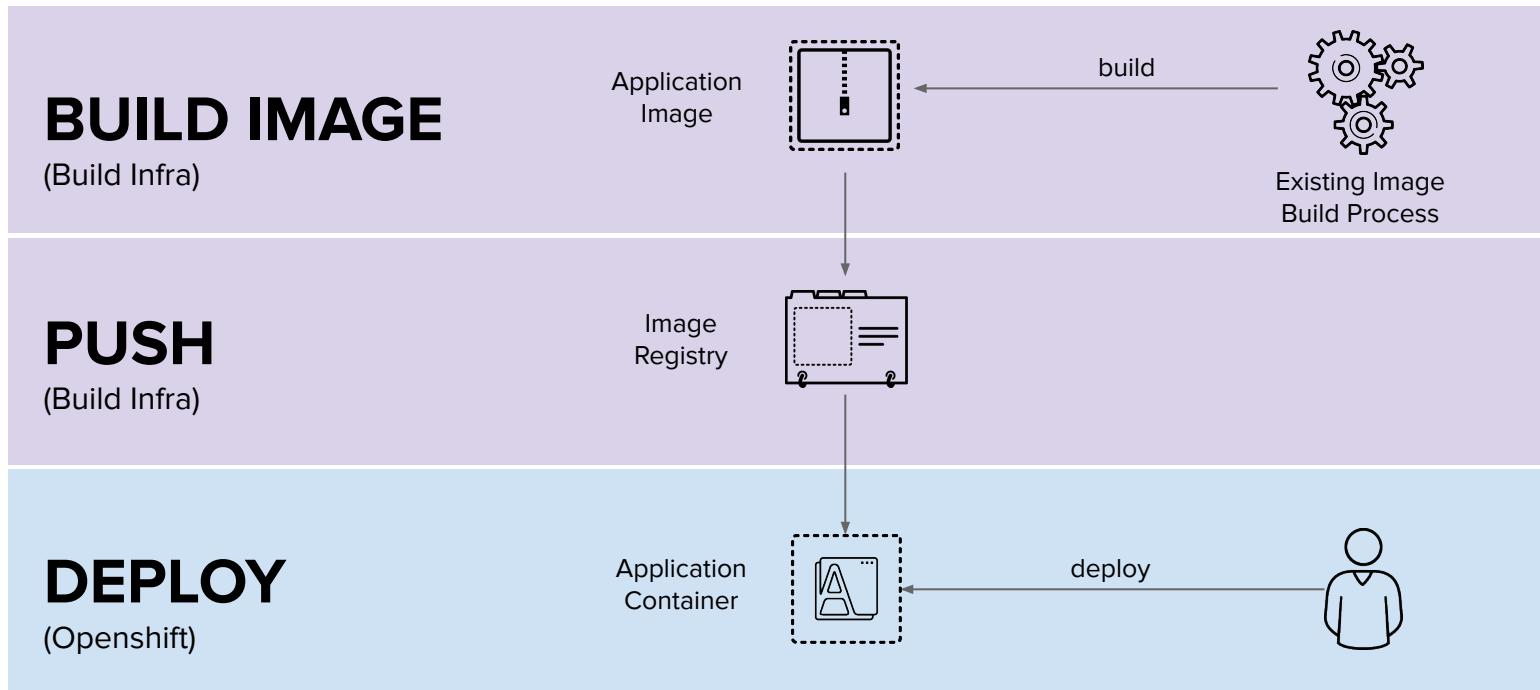
DEPLOY SOURCE CODE WITH SOURCE-TO-IMAGE (S2I)



DEPLOY APP BINARY WITH SOURCE-TO-IMAGE (S2I)



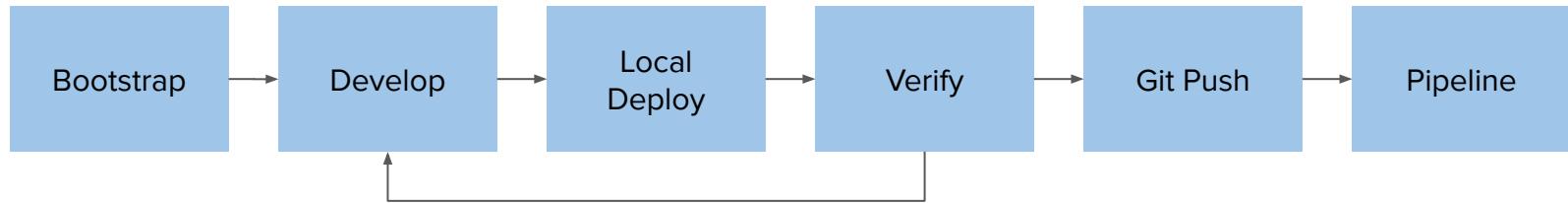
DEPLOY DOCKER IMAGE



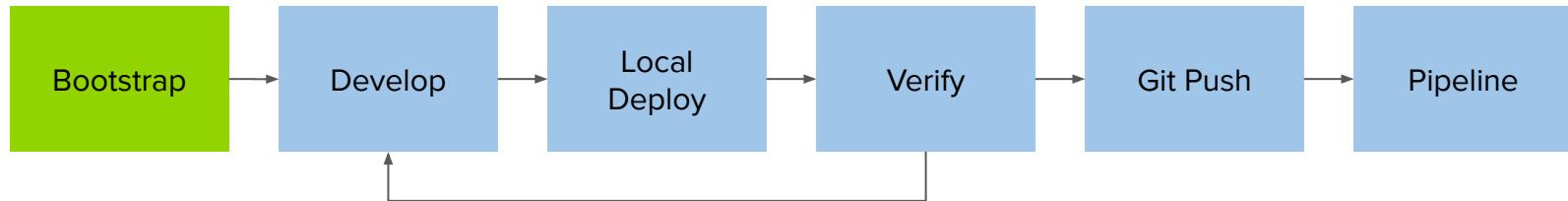
User/Tool Does OpenShift Does

DEVELOPER WORKFLOW

LOCAL DEVELOPMENT WORKFLOW



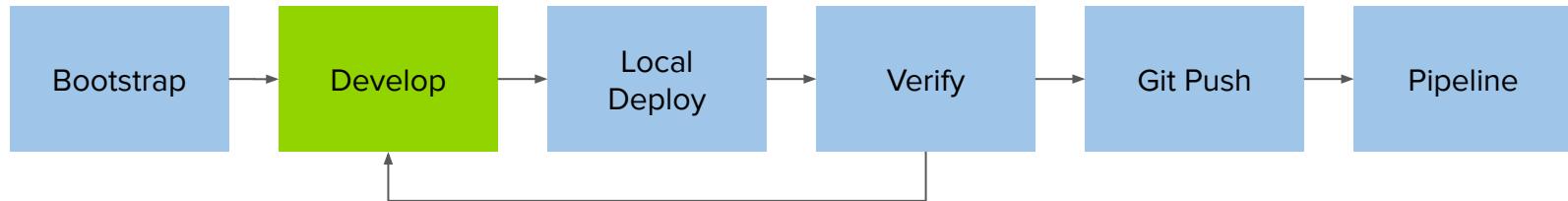
LOCAL DEVELOPMENT WORKFLOW



BOOTSTRAP

- Pick your programming language and application runtime of choice
- Create the project skeleton from scratch or use a generator such as
 - Maven archetypes
 - Quickstarts and Templates
 - OpenShift Generator
 - Spring Initializr

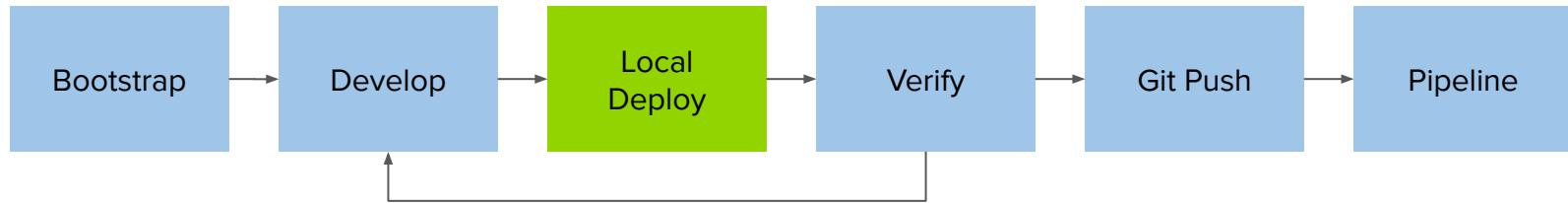
LOCAL DEVELOPMENT WORKFLOW



DEVELOP

- Pick your framework of choice such as Java EE, Spring, Ruby on Rails, Django, Express, ...
- Develop your application code using your editor or IDE of choice
- Build and test your application code locally using your build tools
- Create or generate OpenShift templates or Kubernetes objects

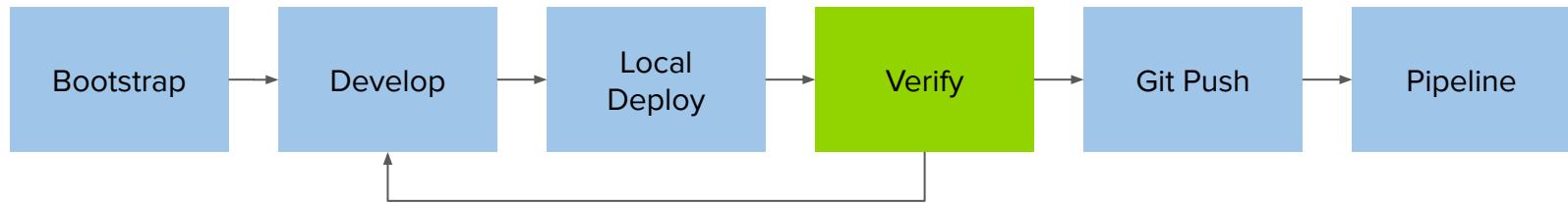
LOCAL DEVELOPMENT WORKFLOW



LOCAL DEPLOY

- Deploy your code on a local OpenShift cluster
 - Red Hat Container Development Kit (CDK), minishift and oc cluster
- Red Hat CDK provides a standard RHEL-based development environment
- Use binary deploy, maven or CLI rsync to push code or app binary directly into containers

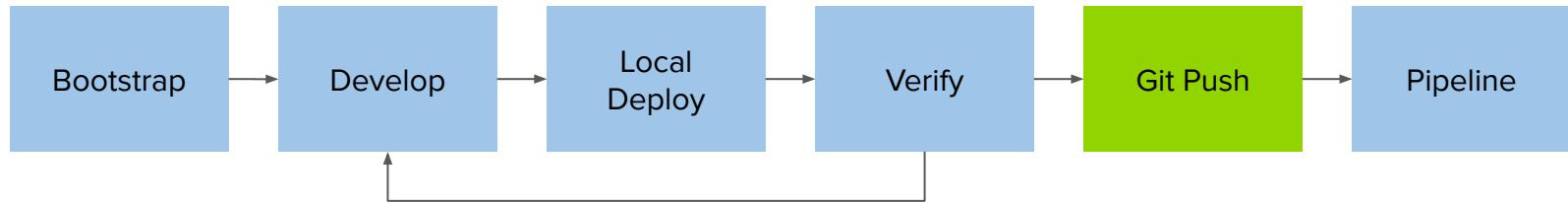
LOCAL DEVELOPMENT WORKFLOW



VERIFY

- Verify your code is working as expected
- Run any type of tests that are required with or without other components (database, etc)
- Based on the test results, change code, deploy, verify and repeat

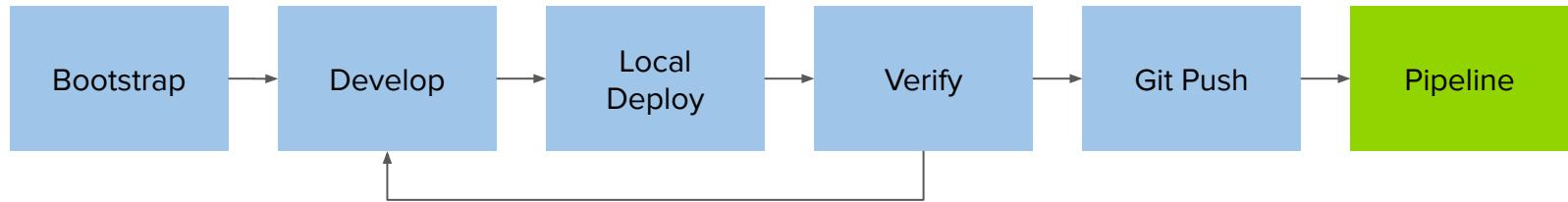
LOCAL DEVELOPMENT WORKFLOW



GIT PUSH

- Push the code and configuration to the Git repository
- If using Fork & Pull Request workflow, create a Pull Request
- If using code review workflow, participate in code review discussions

LOCAL DEVELOPMENT WORKFLOW

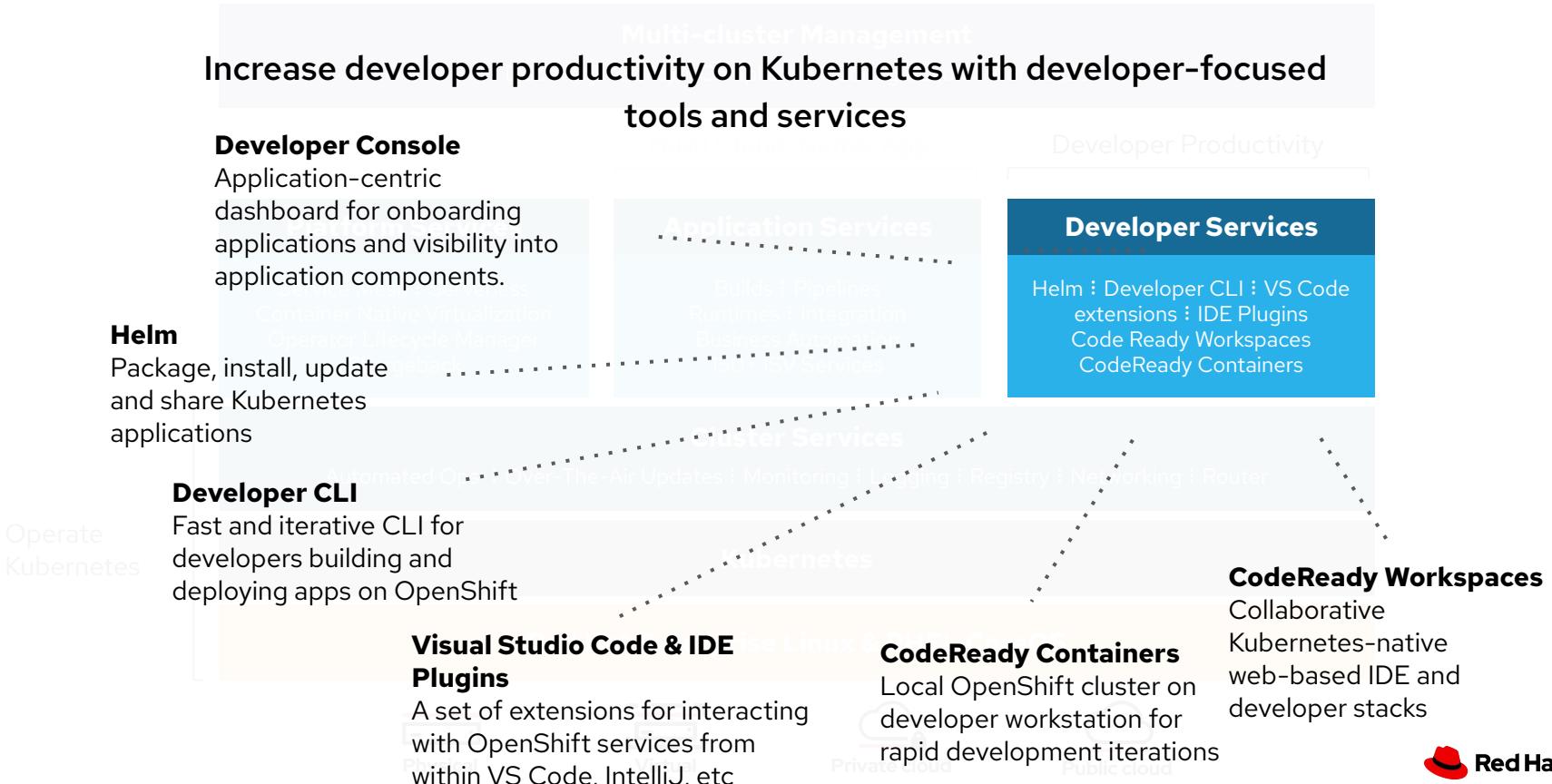


PIPELINE

- Pushing code to the Git repository triggers one or multiple deployment pipelines
- Design your pipelines based on your development workflow e.g. test the pull request
- Failure in the pipeline? Go back to the code and start again

OpenShift Developer Ecosystem

OpenShift Container Platform



Red Hat Runtimes

- Quarkus GA - Supersonic, Subatomic Java.
Native support coming in Q3 via Mandrel.
- Red Hat SSO - Support for OpenShift secret Vaults, WebAuthn protocol.
- Data Grid 8.1 - Cross-site cluster support
- Z-Series support for Runtimes - Use existing Z & Cloud Pak investment with Runtimes & OpenShift.
- JBOSS EAP expansion pack for MicroProfile
- Spring Boot 2.2 - New AMQ Starters, GA of Reactive support and Kubernetes Java annotations.

[What's New in Red Hat Runtimes \[Q2\]](#)

LAUNCH SERVICE

CLOUD-NATIVE RUNTIMES

Red Hat
JBoss Enterprise Application Platform

Red Hat
Data Grid

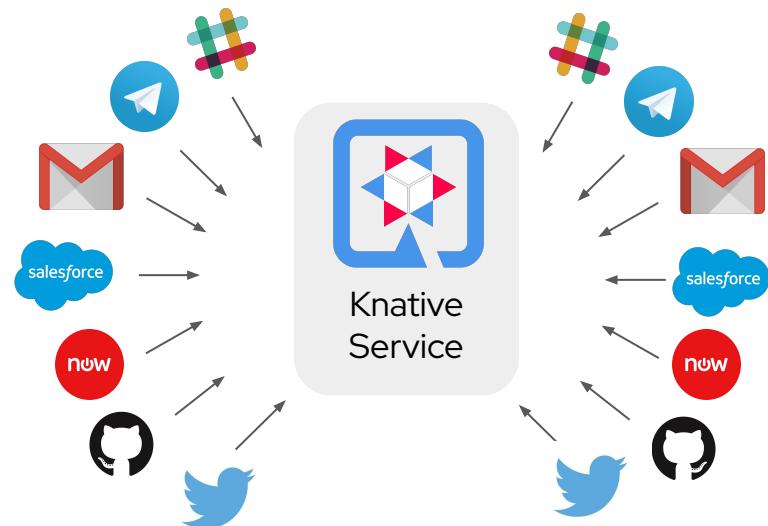
OpenJDK™

RED HAT®
SSO

Red Hat
Application Migration Toolkit

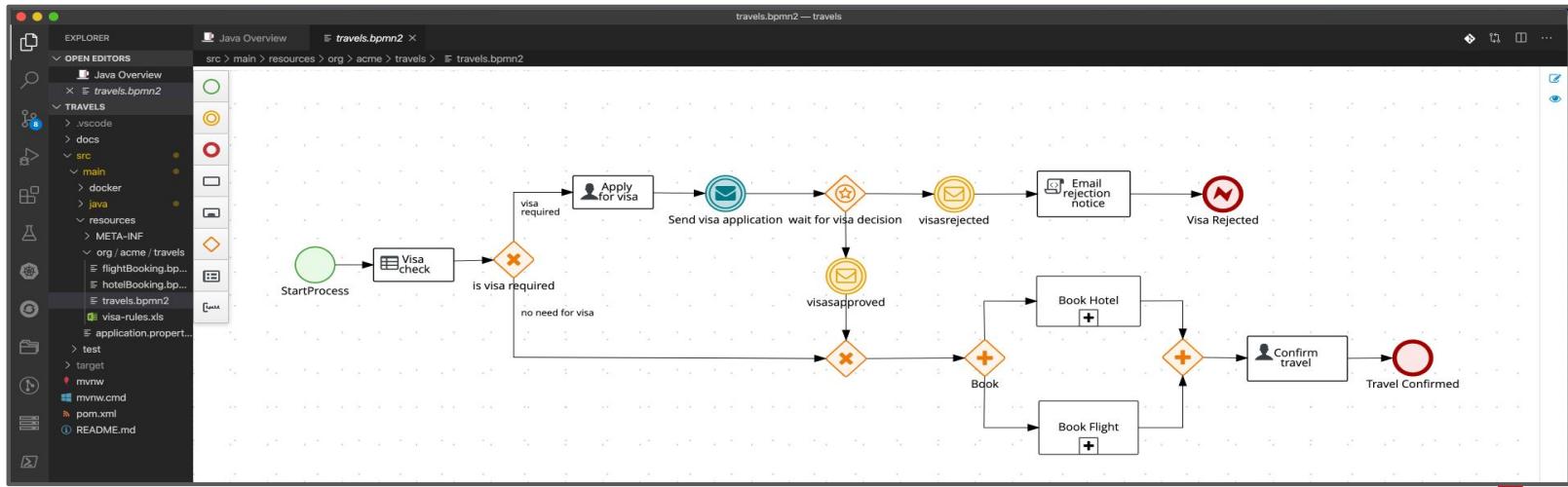
Red Hat Integration

- **Service Registry GA** - schema and API registry to support Kafka and API-based workloads on OpenShift.
- **Mirror Maker 2.0** - major evolution in data replication for Kafka on OpenShift now at full support in Strimzi
- **Camel K for Serverless (TP)** - leverage the huge Camel connector catalog to drive events into your OpenShift Serverless applications based on Camel K and Knative Eventing.



Red Hat Process Automation

- **Kogito** - moves to developer preview for next-gen business automation based on Quarkus
- **Trusty AI** - first MVP of Trusty AI in the next Kogito developer preview release. With runtime dashboards and metrics.
- **Red Hat Business Automation Bundle** - released in the [Visual Studio Code Marketplace](#) as Developer Preview



Quarkus Business Value



QUARKUS

Supersonic. Subatomic. Java.

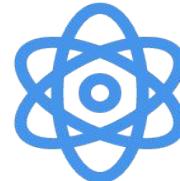
Quarkus - Kubernetes Native Java



Monolith



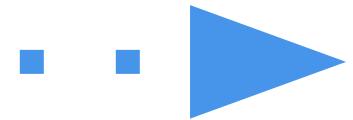
Cloud Native



Microservices



Serverless



Event-Driven
Architecture



kubernetes



Istio



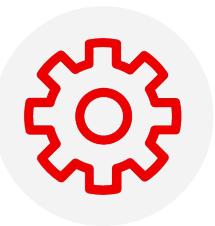
Knative

WHAT IS QUARKUS?

QUARK: elementary particle / **US**: hardest thing in computer science

Quarkus Business Value

"Supersonic, Subatomic Java"



Cost Savings

Cloud efficiency (low memory, fast startup, high density), serverless deployments



Faster Time to Market

Developer productivity, extensions ecosystem, low learning curve, keep competitive edge



Reliability

Trusted technology, active community, trusted sponsor, proven Java libraries and standards

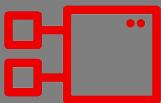
Why Quarkus?

"Quarkus is a great fit if you are looking for..."



IMPROVED APP PERFORMANCE & COST

- Reduce memory consumption
- Increase startup speeds
- Reduce application size
- Looking for Spring or Java alternatives



DIGITAL TRANSFORMATION

- Modernize or optimize existing applications
- Move to a cloud-native or microservices architecture
- Need lightweight connectivity between services
- Next generation business automation



SERVERLESS

- Have seldom used but critical services
- Interested in serverless or FaaS (function as a service)

Quarkus Technical Value

“Historical” Enterprise Java Stack

Architecture: **Monoliths**



Deployment: **multi-app, appserver**

Dynamic Application Frameworks

App Lifecycle: **Months**

Application Server

Memory: **1GB+ RAM**

Java Virtual Machine (Hotspot)

Startup Time: **10s of sec**

Operating System + Hardware/VM

“Modern” Enterprise Java Stack

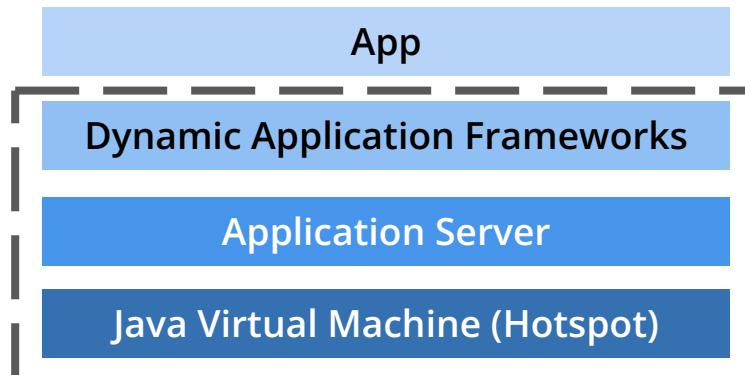
Architecture: **Microservices**

Deployment: **Single App**

App Lifecycle: **Days**

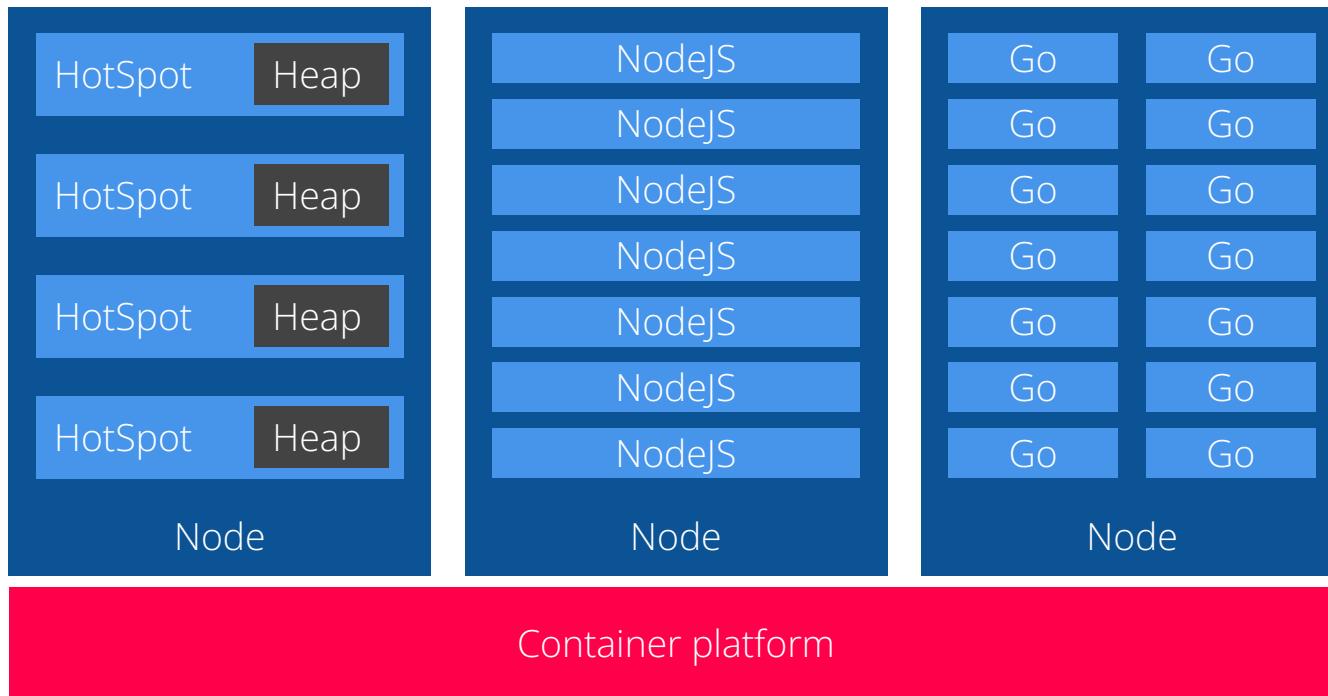
Memory: **100MBs+ RAM**

Startup Time: **Seconds**



No
Change

Hidden Truth About Java + Containers



**THERE IS A NEED FOR A
NEW JAVA STACK FOR
CLOUD-NATIVE AND
SERVERLESS**

Experts from cloud-native Java OS projects

VERT.X



Eclipse Vert.x



Hibernate



RESTEasy



Eclipse MicroProfile



Undertow

OpenJDK™



QUARKUS

Differentiators



Container First

- Tailors your app for HotSpot & GraalVM
- Fast boot time and low RSS memory
- Serverless fit



Developer Joy

- Live coding
- Unified configuration



Unifies Imperative & Reactive

- Combines blocking and non-blocking
- Built-in event bus



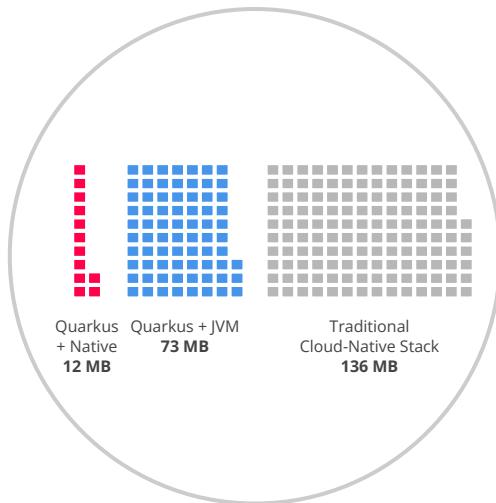
Best of Breed Libraries & Standards

- 90+ extensions
- "Powered by Quarkus" applications

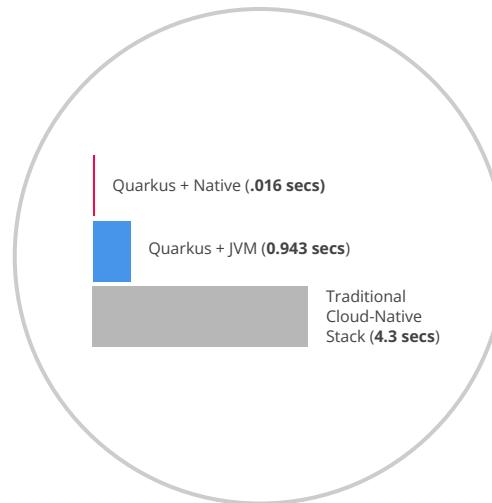
Benefit No. 1: Container First

"We went from 1-min startup times to 400 milliseconds"

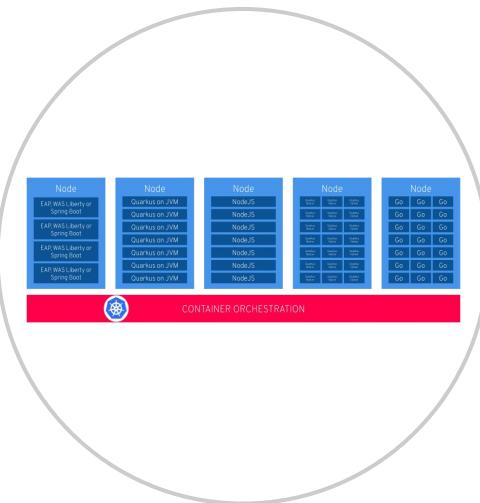
Reduced Memory Footprint



Fast Startup Time



Smaller Disk Footprint

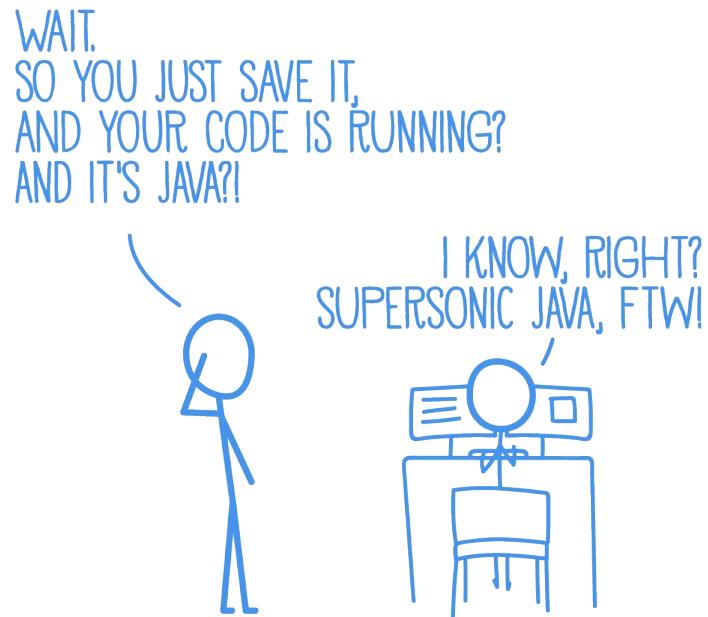


Benefit No. 2: Developer Joy

*"Our developers used to wait **2 to 3 mins** to see their changes. **Live coding** does away with this."*

A cohesive platform for optimized developer joy:

- Based on standards and more
- Unified configuration
- Live coding
- Streamlined code for the 80% common usages, flexible for the 20%
- No hassle native executable generation



Benefit No. 3: Unifies Imperative and Reactive

```
@Inject  
SayService say;  
  
@GET  
@Produces(MediaType.TEXT_PLAIN)  
public String hello() {  
    return say.hello();  
}
```

```
@Inject @Stream("kafka")  
Publisher<String> reactiveSay;  
  
@GET  
@Produces(MediaType.SERVER_SENT_EVENTS)  
public Publisher<String> stream() {  
    return reactiveSay;  
}
```

- Combine both Reactive and imperative development in the same application
- Inject the EventBus or the Vertx context
- Use the technology that fits your use-case
- Key for reactive systems based on event driven apps

Benefit No. 4: Best of Breed Frameworks & Standards

"When you adopt Quarkus, you will be productive from day one since you don't need to learn new technologies."



Eclipse Vert.x



Hibernate



RESTEasy



Apache Camel



Eclipse MicroProfile



Netty



Kubernetes



OpenShift



Jaeger



Prometheus



Apache Kafka



Infinispan



Flyway



Neo4j



MongoDB



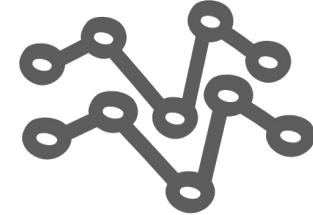
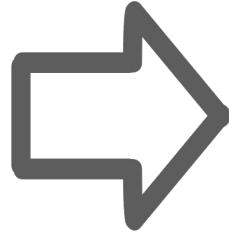
MQTT



Keycloak



Apache Tika



Supersonic, Subatomic

Fast.

Blazing fast to start.

Millisecond fast!

Supersonic, Subatomic Java

REST

Quarkus + Native (via GraalVM) **0.016 Seconds**

Quarkus + JVM (via OpenJDK) **0.943 Seconds**

Traditional Cloud-Native Stack **4.3 Seconds**

REST + CRUD

Quarkus + Native (via GraalVM) **0.042 Seconds**

Quarkus + JVM (via OpenJDK) **2.033 Seconds**

Traditional Cloud-Native Stack **9.5 Seconds**

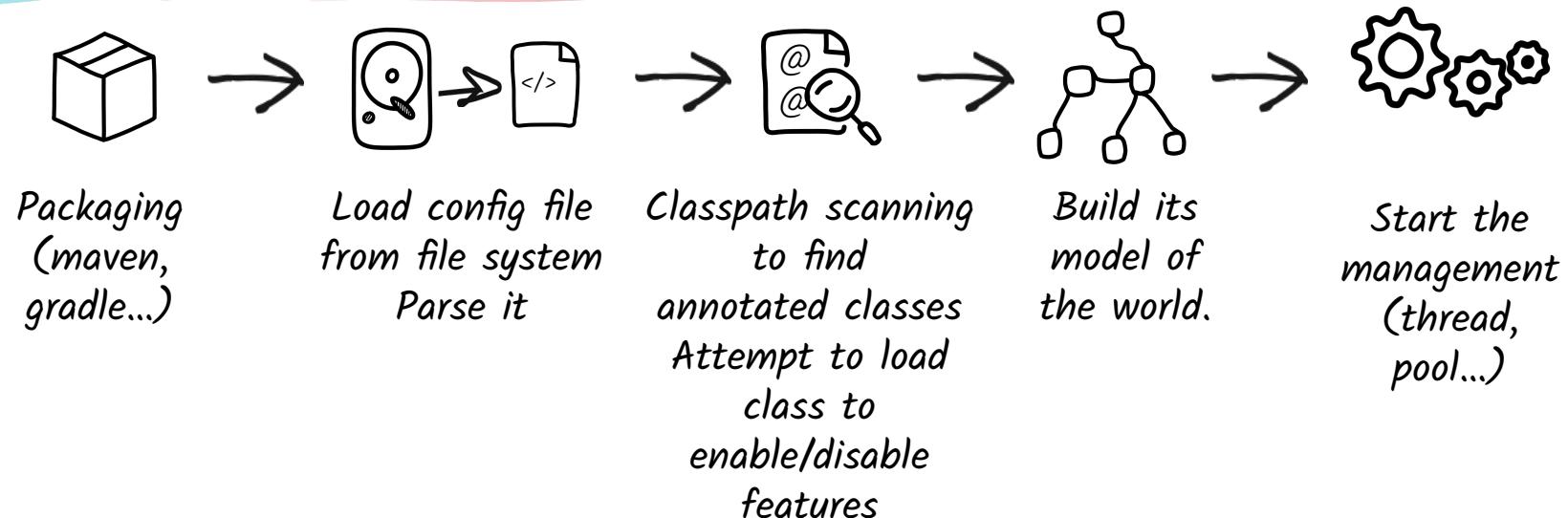
Time to first response

HOW DOES QUARKUS WORK?

How Does a Framework Start?

Build Time

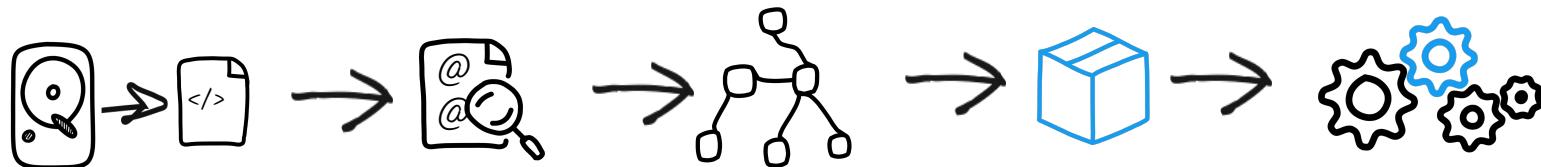
Runtime



The Quarkus Way

Build Time

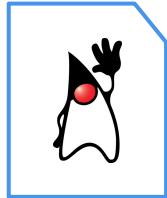
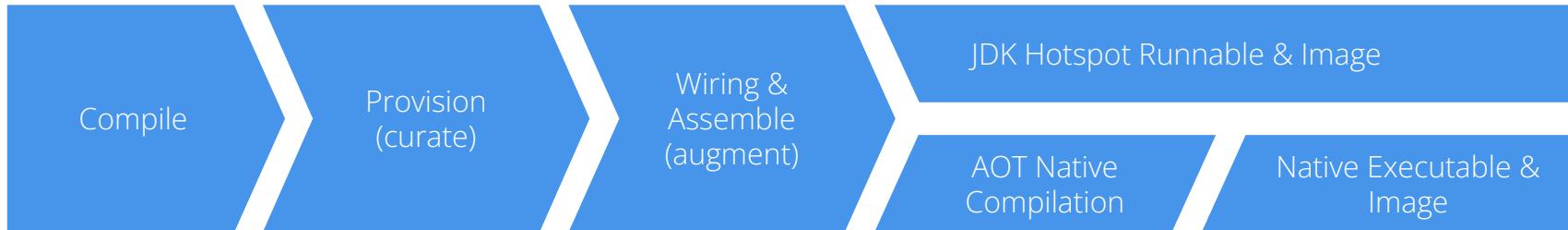
Runtime



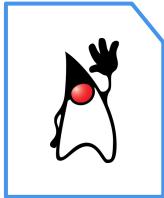
Build Time

Runtime

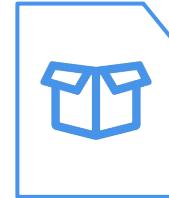
An ahead-of-time, build-time, runtime



app.jar



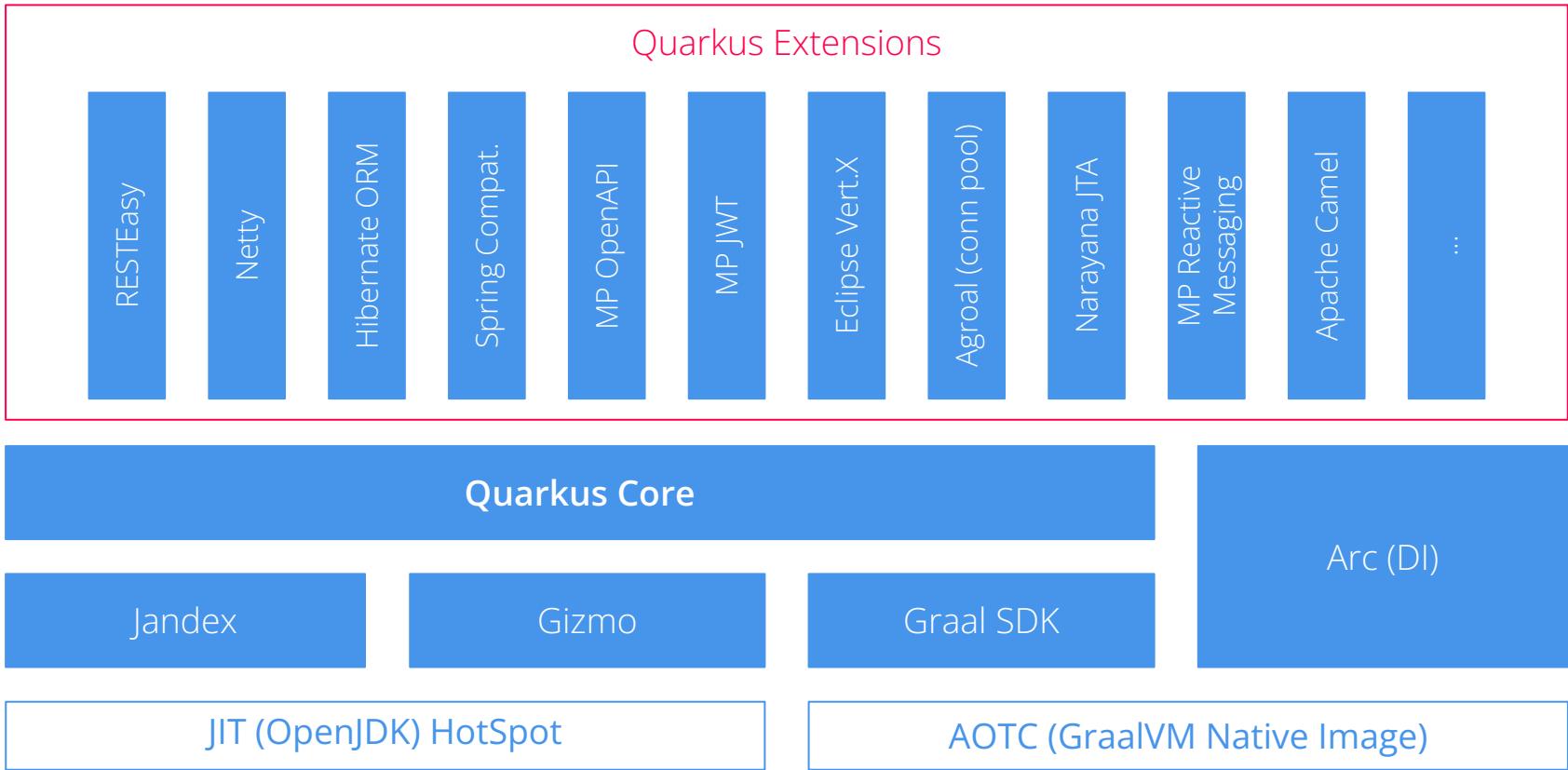
frameworks



Runnable java app



native-app



The Right VM For the Right Deployment

JIT (OpenJDK HotSpot)

- High memory density requirements
- High request/s/MB
- Fast startup time
- Best raw performance (CPU)
- Best garbage collectors
- Higher heap size usage
- Known monitoring tools
- Compile Once, Run anywhere
- Libraries that only work in standard JDK

AOT (GraalVM native image)*

- Highest memory density requirements
- Highest request/s/MB
for low heap size usages
- Faster startup time
10s of ms for Serverless

*Currently in Tech Preview



Quarkus Tools - Build

maven



Gradle*

```
mvn io.quarkus:quarkus-maven-plugin:1.3.2.Final-redhat-00001:create \
-DprojectGroupId=org.acme \
-DprojectArtifactId=getting-started \
-DplatformGroupId=com.redhat.quarkus \
-DplatformVersion=1.3.2.Final-redhat-00001 \
-DclassName="org.acme.quickstart.GreetingResource" \
-Dpath="/hello"
cd getting-started
```

*community supported



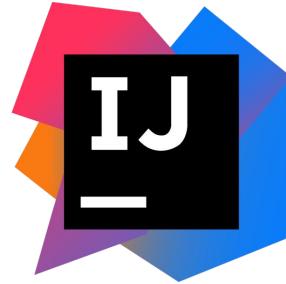
Quarkus Tools - IDE



[VSCode](#)



[Eclipse](#)



[IntelliJ](#)



[che.openshift.io](#)

DEMO

Please be back at 11:40 CET

THANK YOU