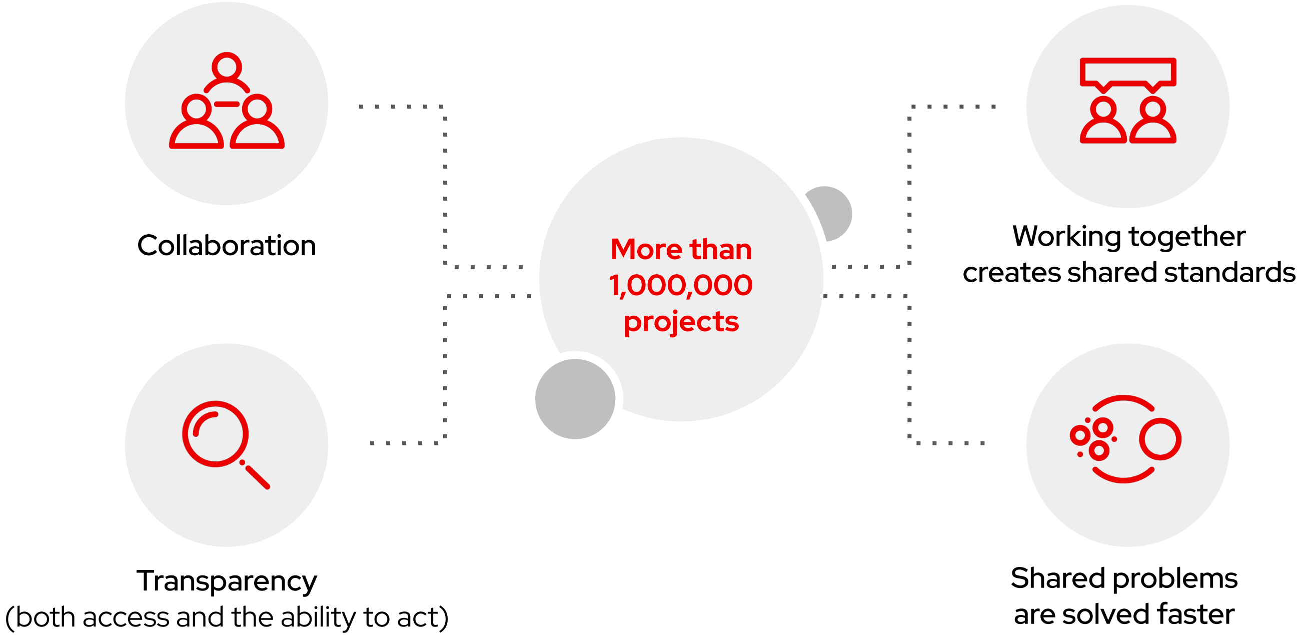
OPENSHIFT CONTAINER PLATFORM 4

“Developers, Developers, Developers…”

Florian Moss

Solution Architect, Red Hat

Open source culture



More than 1,000,000 projects

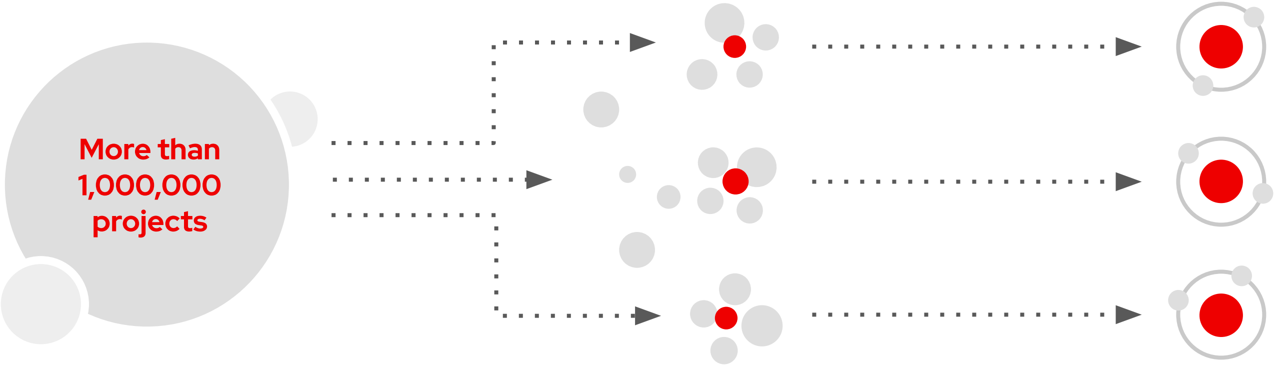
Collaboration

Transparency
(both access and the ability to act)

Working together
creates shared standards

Shared problems
are solved faster

Product development model



Participate

We participate in and create community-powered upstream projects.

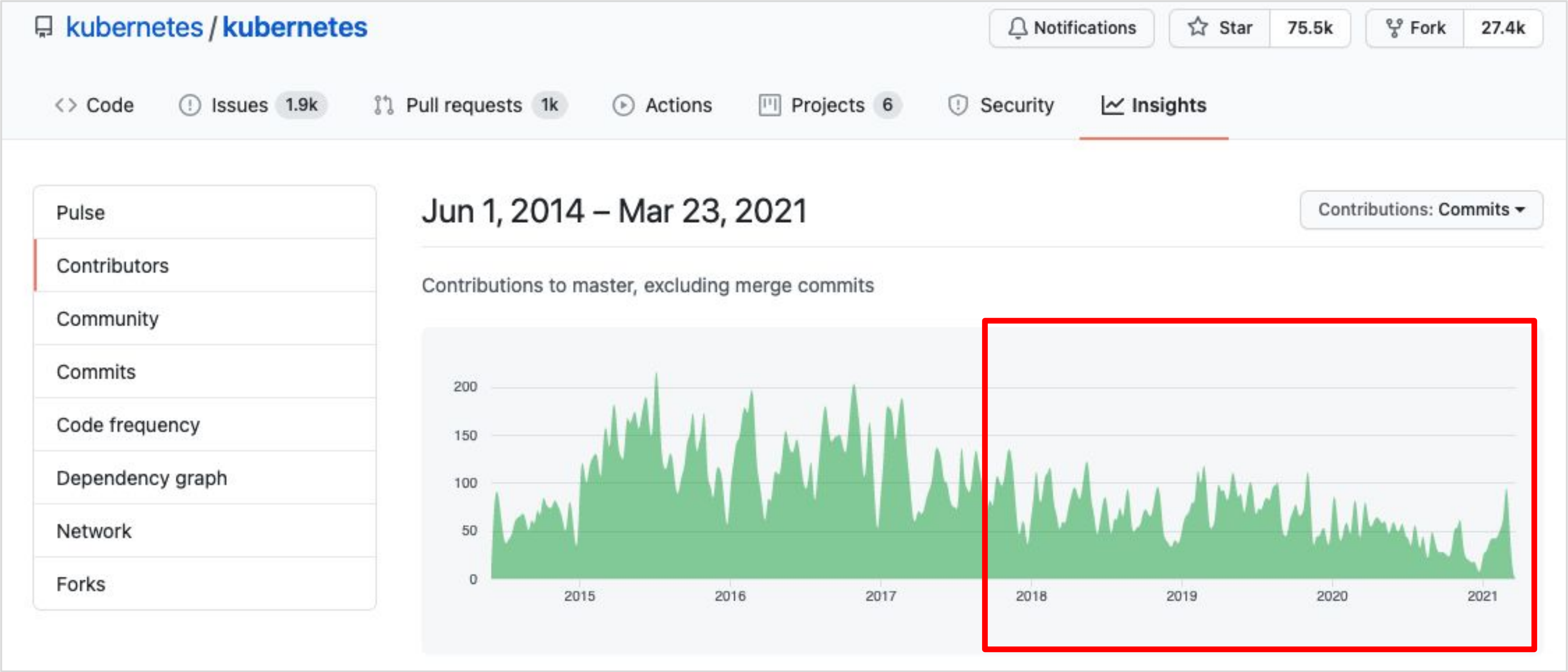
Integrate

We integrate upstream projects, fostering open community platforms.

Stabilize

We commercialize these platforms together with a rich ecosystem of services and certifications.

Kubernetes development is slowing



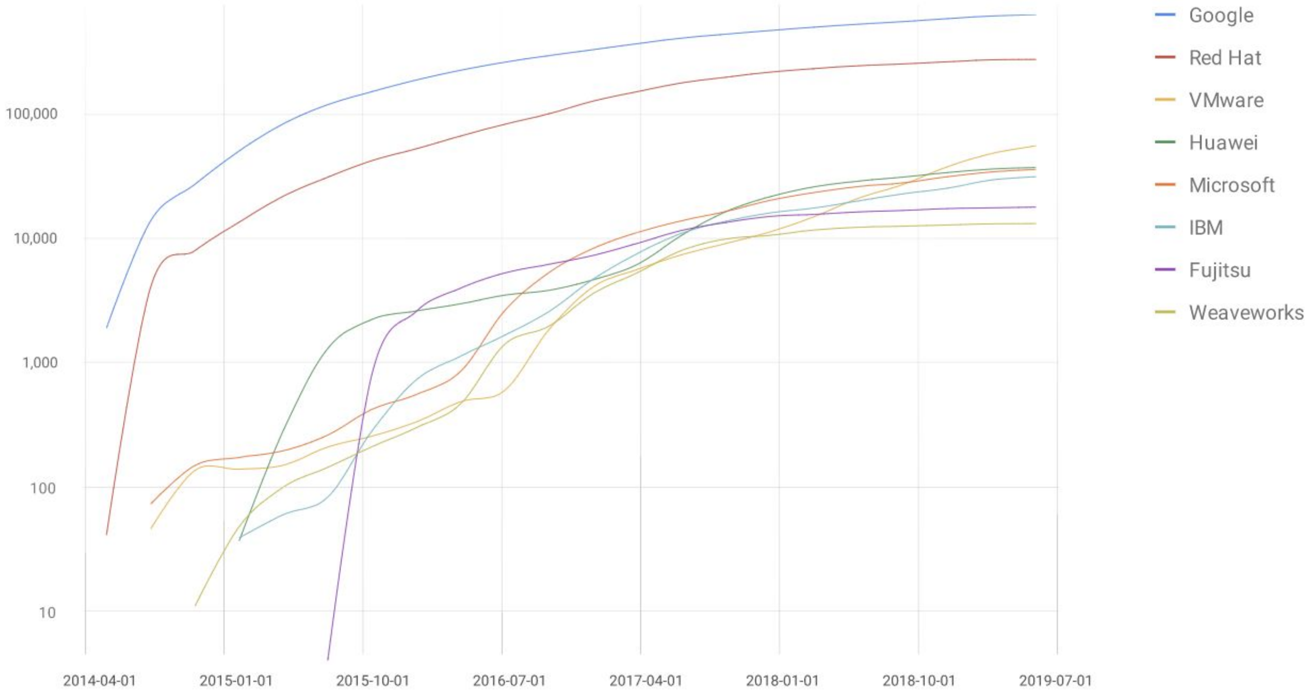
Jun 1, 2014 – Mar 23, 2021

Contributions to master, excluding merge commits

Source: GitHub, “Contributing to Kubernetes”, accessed 17 April 2020.

Sharing stewardship & growing the community

Code diversity: no single company dominates contributions to Kubernetes

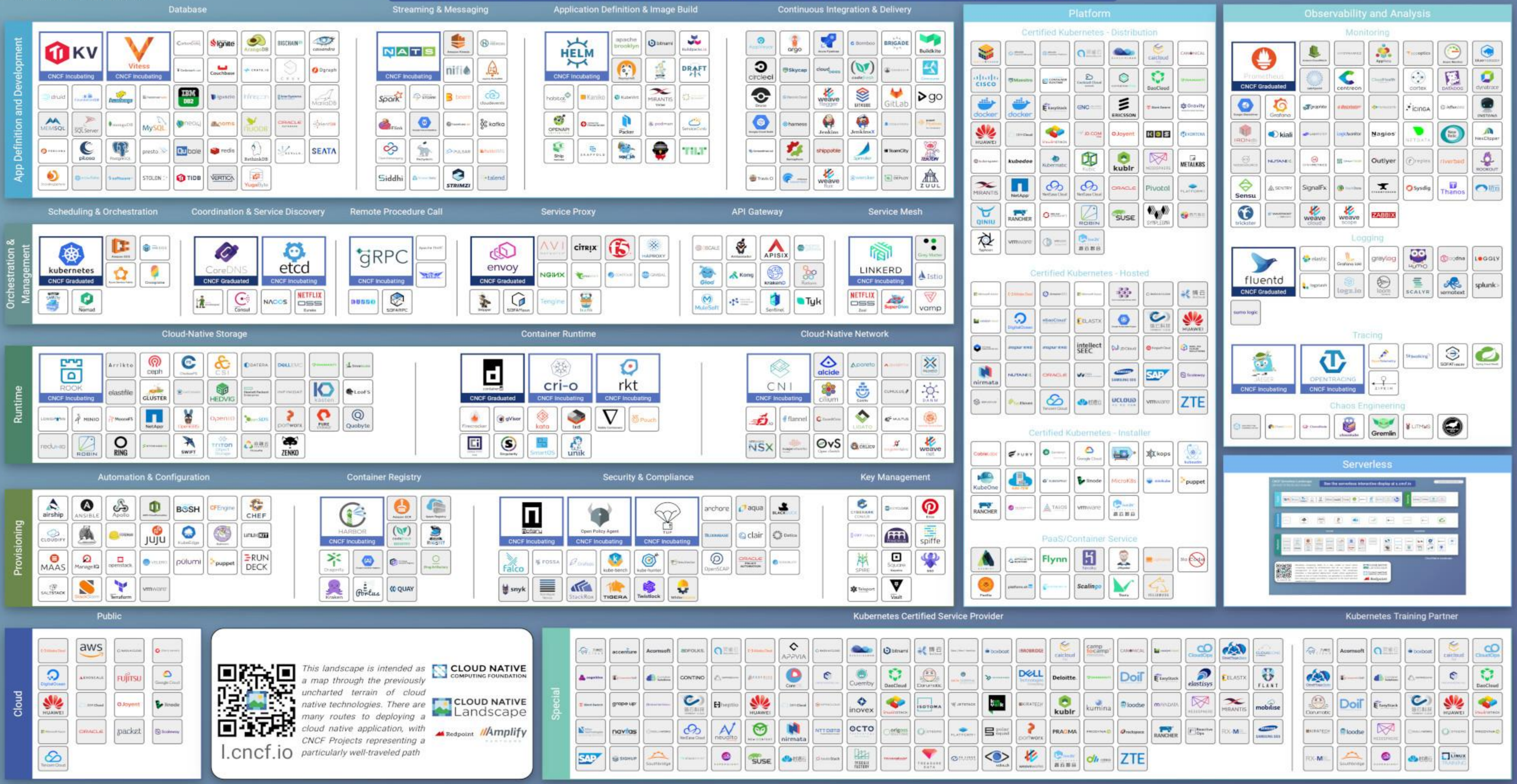


Source: Cloud Native Computing Foundation. “CNCF Kubernetes Project Journey Report.” 2019.

CNCF Cloud Native Landscape
2019-07-12T03:51:02Z 024e93b

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

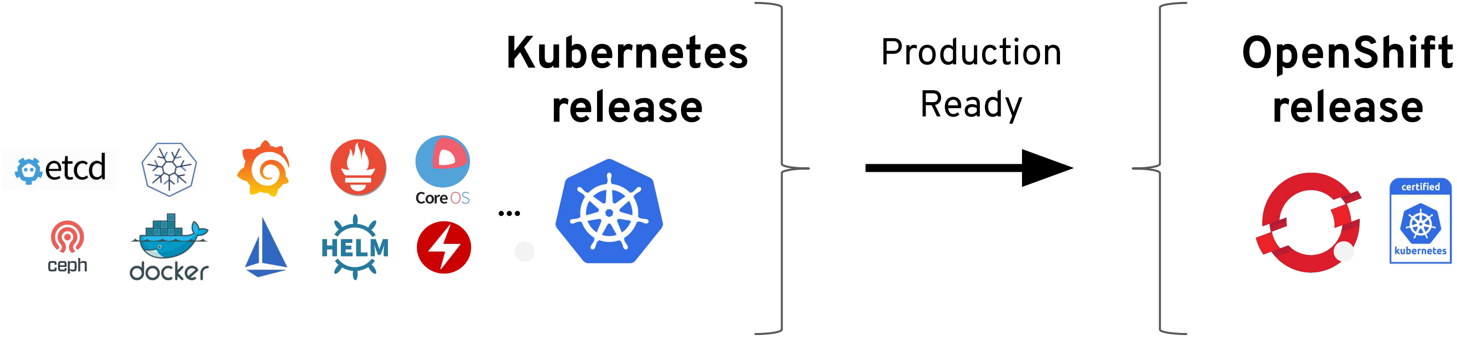
Greyed logos are not open source



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

l.cncf.io

In a nutshell...



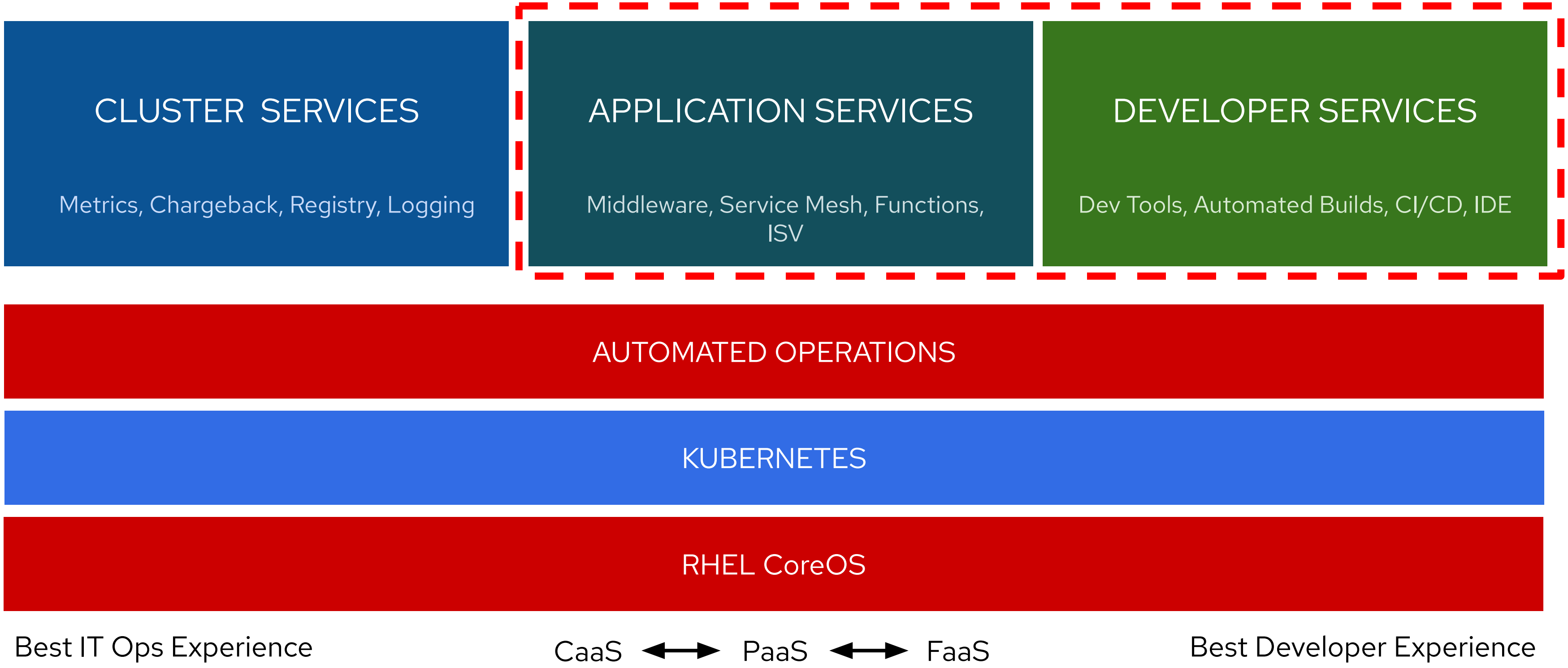
Kubernetes release

Production Ready

OpenShift release

- Hundreds of defect and performance fixes
- 200+ validated integrations
- Certified container ecosystem
- 9-year enterprise life-cycle management
- Red Hat is a leading Kubernetes contributor since day 1

OpenShift 4 Platform



CLUSTER SERVICES

Metrics, Chargeback, Registry, Logging

APPLICATION SERVICES

Middleware, Service Mesh, Functions, ISV

DEVELOPER SERVICES

Dev Tools, Automated Builds, CI/CD, IDE

AUTOMATED OPERATIONS

KUBERNETES

RHEL CoreOS

Best IT Ops Experience

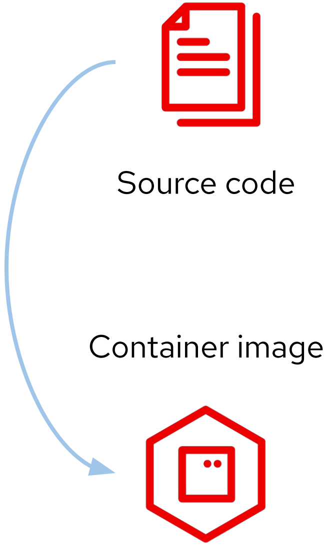
CaaS ⟷ PaaS ⟷ FaaS

Best Developer Experience

OpenShift for Developers

Development team objectives

- Limit what I need to learn
- Create applications quickly and easily
- Fast cycle of edit-build-deploy-test
- View what is going on within an application
- Access to environments without delay
- Build, test, deploy in a repeatable manner
- Create manageable CI/CD processes - Extensive to add testing etc.



Source code

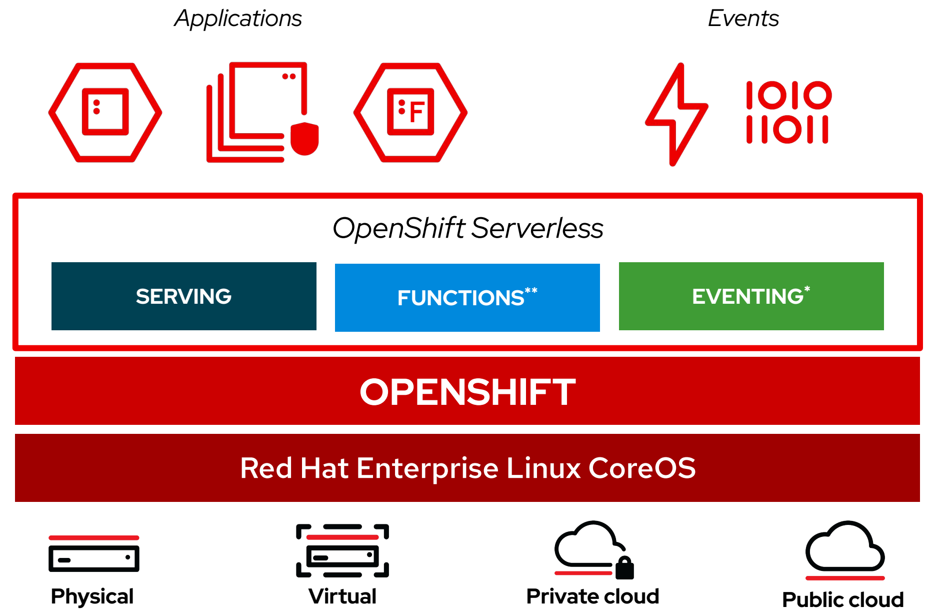
Container image

1 Serverless

OpenShift Serverless

Event-driven serverless containers and functions

- Deploy and run **serverless containers**
- Use any programming language or runtime
- Modernize existing applications to run serverless
- Powered by a rich ecosystem of event sources
- Manage serverless apps natively in Kubernetes
- Based on open source project **kNative**
- Run anywhere OpenShift runs



Applications

Events

OpenShift Serverless

SERVING

FUNCTIONS**

EVENTING*

OPENSHIFT

Red Hat Enterprise Linux CoreOS

Physical

Virtual

Private cloud

Public cloud

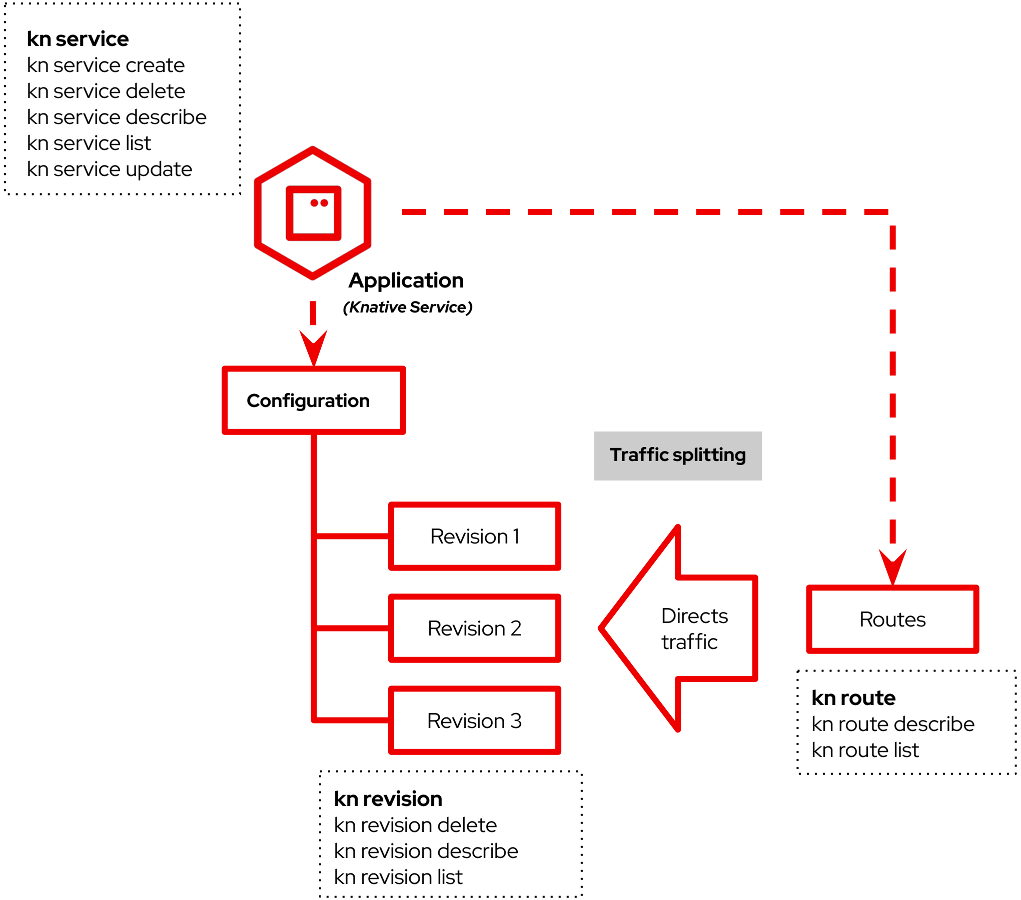
* Eventing is currently in Technology Preview

** Functions are currently a work in progress initiative

Serving

- From container to URL within seconds
- Easier developer experience for Kubernetes
- Built-in versioning, traffic split and more
- Simplified Installation experience with Kourier
- Automatic TLS/SSL for Applications

```
$ kn service create
--image=<container>
```



kn service
kn service create
kn service delete
kn service describe
kn service list
kn service update

Application
(Knative Service)

Configuration

Traffic splitting

Revision 1

Revision 2

Revision 3

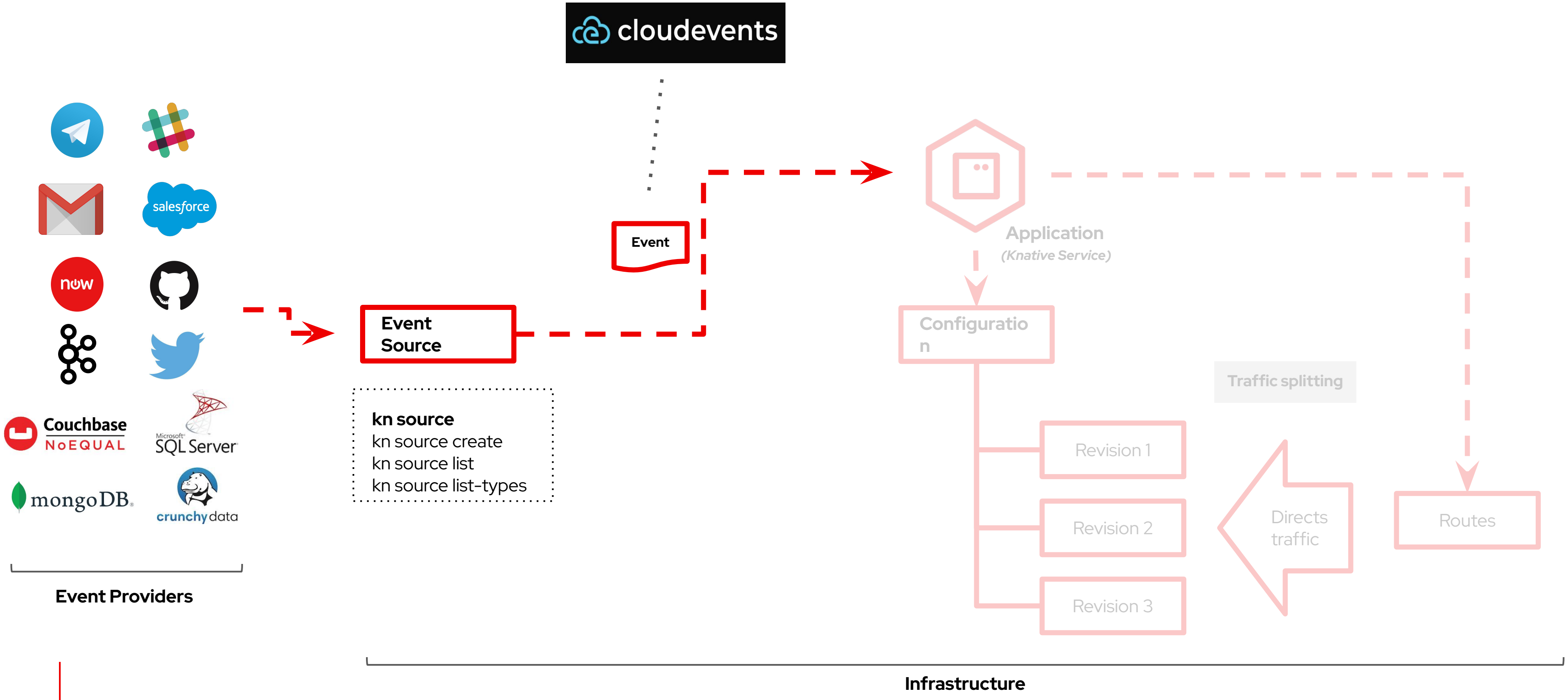
Directs traffic

Routes

kn route
kn route describe
kn route list

kn revision
kn revision delete
kn revision describe
kn revision list

Eventing



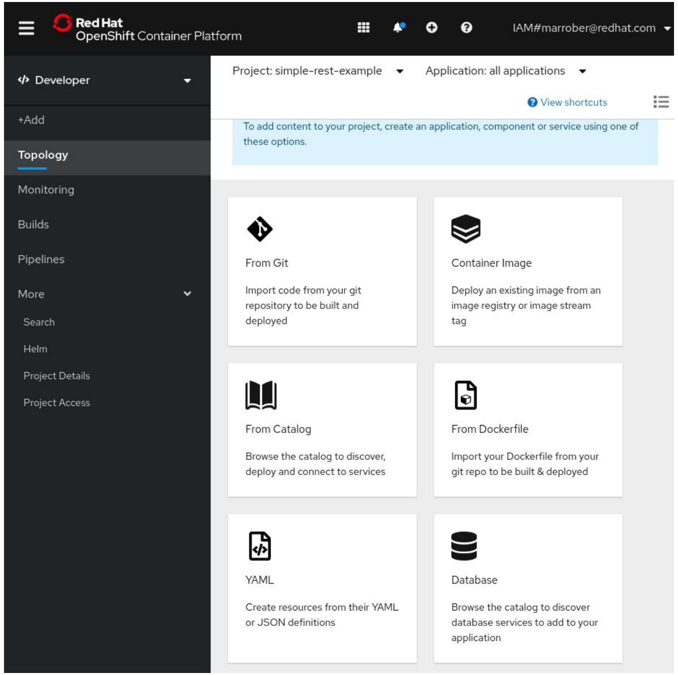
```
kn source
kn source create
kn source list
kn source list-types
```

2 GUI

OpenShift for the developer

Create an application from a variety of sources

- Git repository of source code
- An existing container image stored in a repository
- Catalog item
  - Database, runtime platform, middleware etc.
- Dockerfile
- Resource from YAML file
- Database service



To add content to your project, create an application, component or service using one of these options.

From Git

Import code from your git repository to be built and deployed

Container Image

Deploy an existing image from an image registry or image stream tag

From Catalog

Browse the catalog to discover, deploy and connect to services

From Dockerfile

Import your Dockerfile from your git repo to be built & deployed

YAML

Create resources from their YAML or JSON definitions

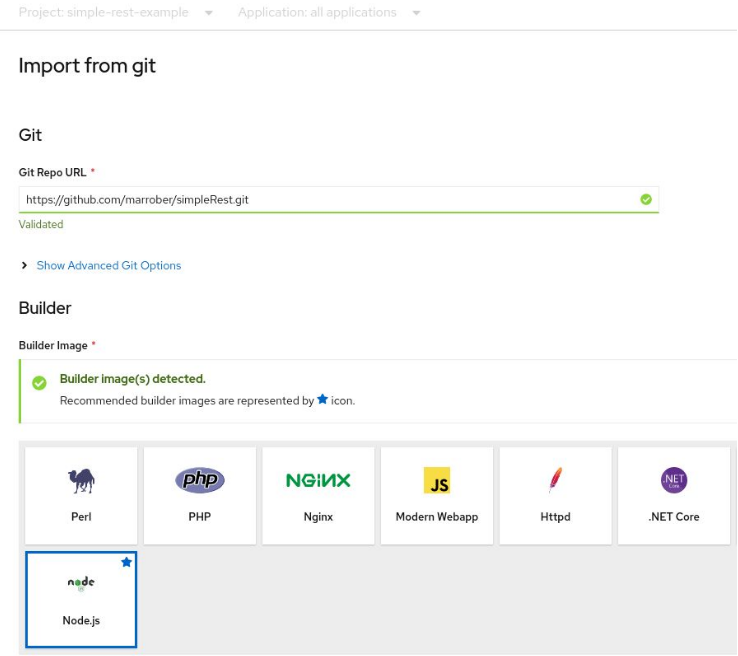
Database

Browse the catalog to discover database services to add to your application

OpenShift for the developer

Git repository with source code

- Git repository of source code
- OpenShift detects the language
- Runtime version selection
- Hit 'Create' – what do we get ...



Project: simple-rest-example ▾ Application: all applications ▾

Import from git

Git

Git Repo URL *

https://github.com/marrober/simpleRest.git ✓

Validated

› Show Advanced Git Options

Builder

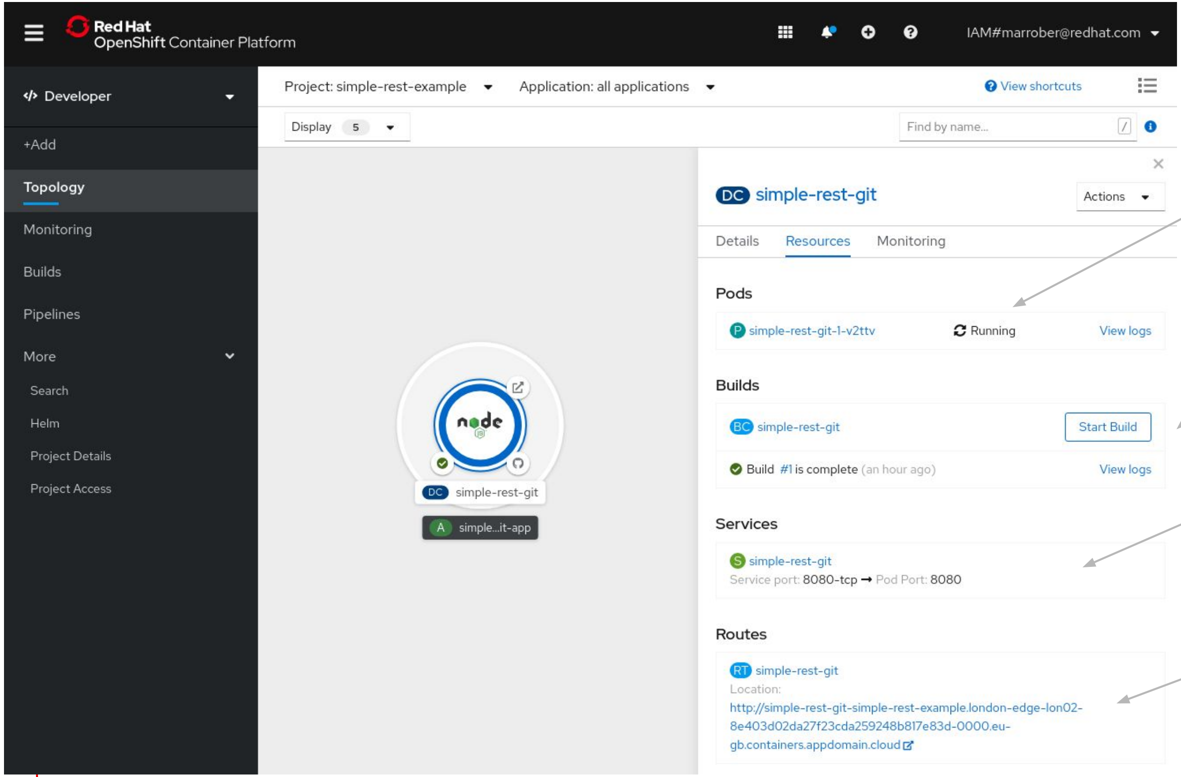
Builder Image *

✓ Builder image(s) detected.

Recommended builder images are represented by ★ icon.

OpenShift for the developer

What gets created ?



DC simple-rest-git

Actions

Details Resources Monitoring

Pods

P simple-rest-git-1-v2ttv Running View logs

Builds

BC simple-rest-git Start Build

Build #1 is complete (an hour ago) View logs

Services

S simple-rest-git

Service port: 8080-tcp → Pod Port: 8080

Routes

RT simple-rest-git

Location:

http://simple-rest-git-simple-rest-example.london-edge-lon02-8e403d02da27f23cda259248b817e83d-0000.eu-gb.containers.appdomain.cloud

Container running in a pod

Source code build task

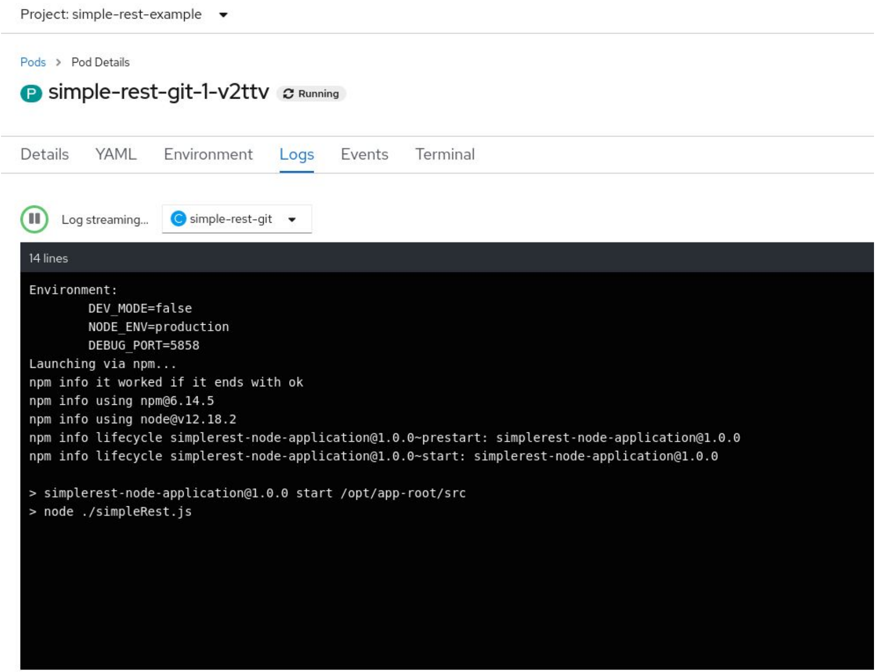
A service to interface to the pod

A route to access the service from outside the cluster

OpenShift for the developer

See more detail of the running pod

- Application log files
- Can also use Elasticsearch for multi-app logging

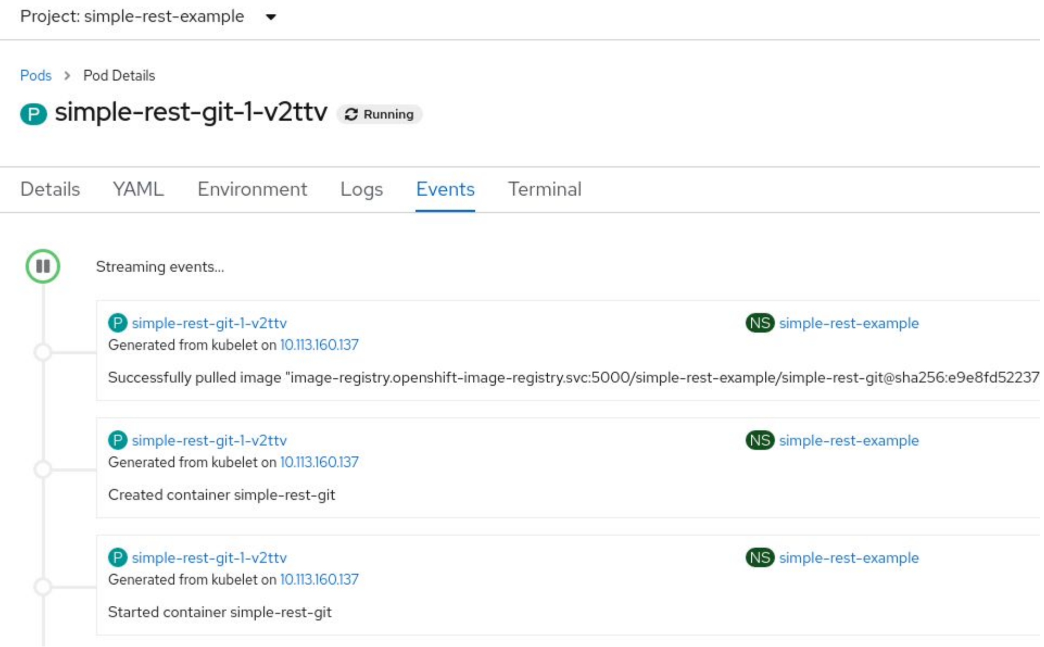


```
Environment:
        DEV_MODE=false
        NODE_ENV=production
        DEBUG_PORT=5858
Launching via npm...
npm info it worked if it ends with ok
npm info using npm@6.14.5
npm info using node@v12.18.2
npm info lifecycle simplerest-node-application@1.0.0~prestart: simplerest-node-application@1.0.0
npm info lifecycle simplerest-node-application@1.0.0~start: simplerest-node-application@1.0.0

> simplerest-node-application@1.0.0 start /opt/app-root/src
> node ./simpleRest.js
```

OpenShift for the developer

See more detail of the running pod

- Events view
  - Probes firing
  - Volumes attaching
  - Pod creation
  - Issues ...



Project: simple-rest-example

Pods > Pod Details

P simple-rest-git-1-v2ttv Running

Streaming events...

P simple-rest-git-1-v2ttv NS simple-rest-example
Generated from kubelet on 10.113.160.137
Successfully pulled image "image-registry.openshift-image-registry.svc:5000/simple-rest-example/simple-rest-git@sha256:e9e8fd52237

P simple-rest-git-1-v2ttv NS simple-rest-example
Generated from kubelet on 10.113.160.137
Created container simple-rest-git

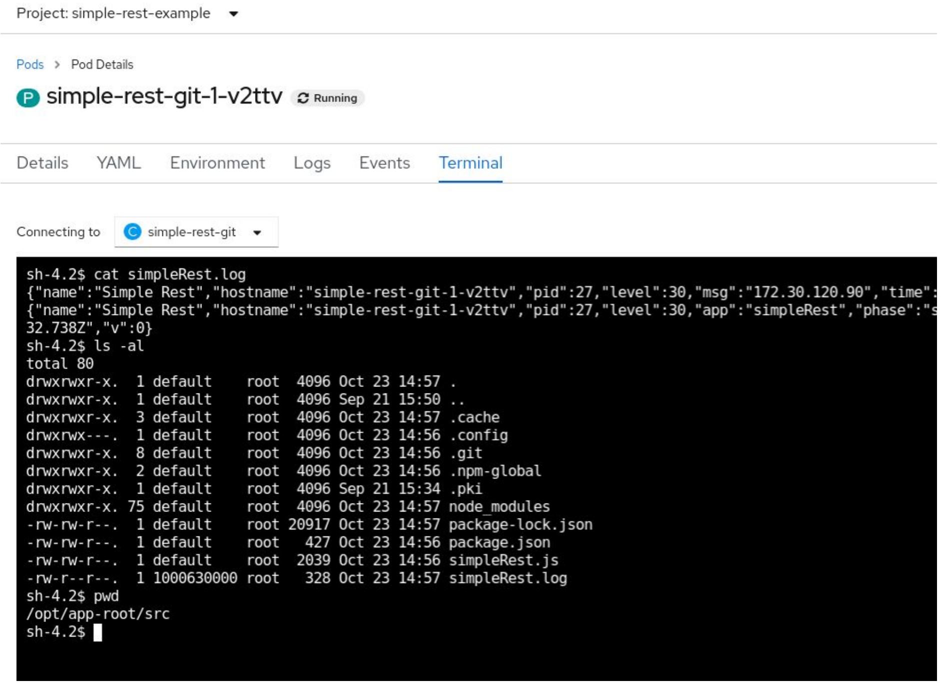
P simple-rest-git-1-v2ttv NS simple-rest-example
Generated from kubelet on 10.113.160.137
Started container simple-rest-git

Red Hat

OpenShift for the developer

See more detail of the running pod

- Terminal into each running container
- No more remote logging into apps

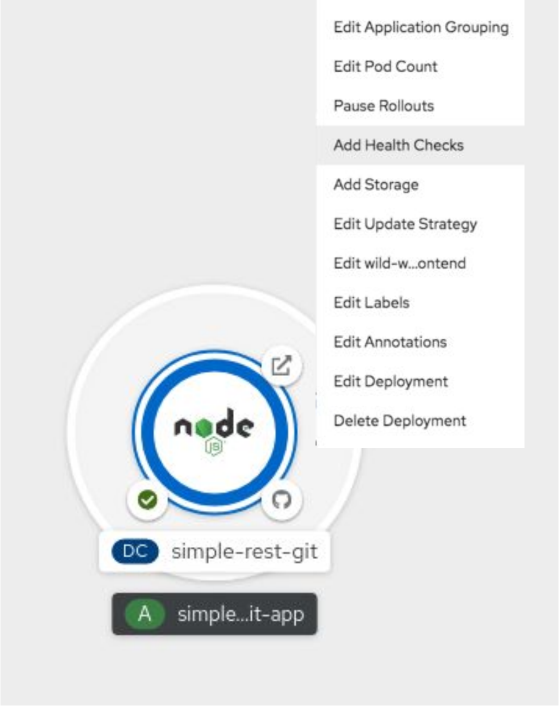


```
sh-4.2$ cat simpleRest.log
{"name":"Simple Rest","hostname":"simple-rest-git-1-v2ttv","pid":27,"level":30,"msg":"172.30.120.90","time":
{"name":"Simple Rest","hostname":"simple-rest-git-1-v2ttv","pid":27,"level":30,"app":"simpleRest","phase":"s
32.738Z","v":0}
sh-4.2$ ls -al
total 80
drwxrwxr-x.  1 default    root  4096 Oct 23 14:57 .
drwxrwxr-x.  1 default    root  4096 Sep 21 15:50 ..
drwxrwxr-x.  3 default    root  4096 Oct 23 14:57 .cache
drwxrwx---.  1 default    root  4096 Oct 23 14:56 .config
drwxrwxr-x.  8 default    root  4096 Oct 23 14:56 .git
drwxrwxr-x.  2 default    root  4096 Oct 23 14:56 .npm-global
drwxrwxr-x.  1 default    root  4096 Sep 21 15:34 .pki
drwxrwxr-x. 75 default    root  4096 Oct 23 14:57 node_modules
-rw-rw-r--.  1 default    root 20917 Oct 23 14:57 package-lock.json
-rw-rw-r--.  1 default    root   427 Oct 23 14:56 package.json
-rw-rw-r--.  1 default    root  2039 Oct 23 14:56 simpleRest.js
-rw-r--r--.  1 1000630000 root   328 Oct 23 14:57 simpleRest.log
sh-4.2$ pwd
/opt/app-root/src
sh-4.2$ 
```

OpenShift for the developer

Manage the application from the topology view

- Create health probes
  - Manage liveness and readiness of applications
  - Take corrective action as needed
- Add storage
- Scale the pods



Monitoring application health

Liveness – Is the container running correctly ?

Recovery Process

Restart the container

Readiness – Is the container ready to service requests ?

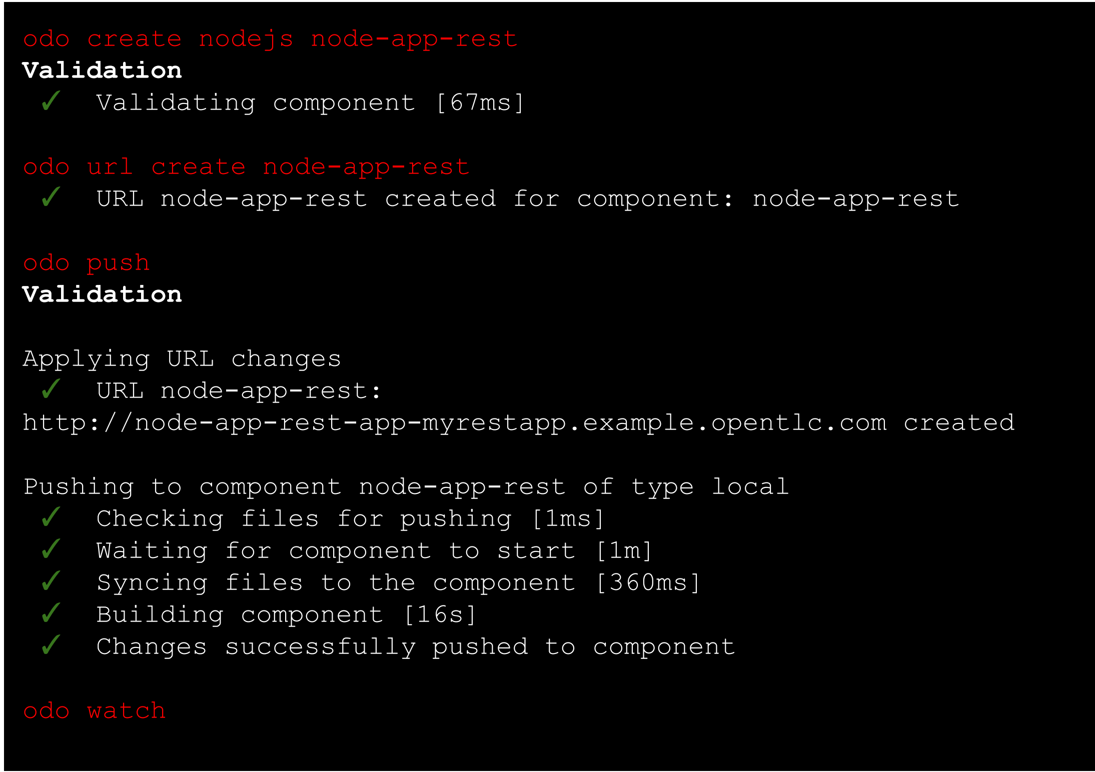
Recovery process

Leave the container running and fix it / wait for other condition

3 Git for Kubernetes

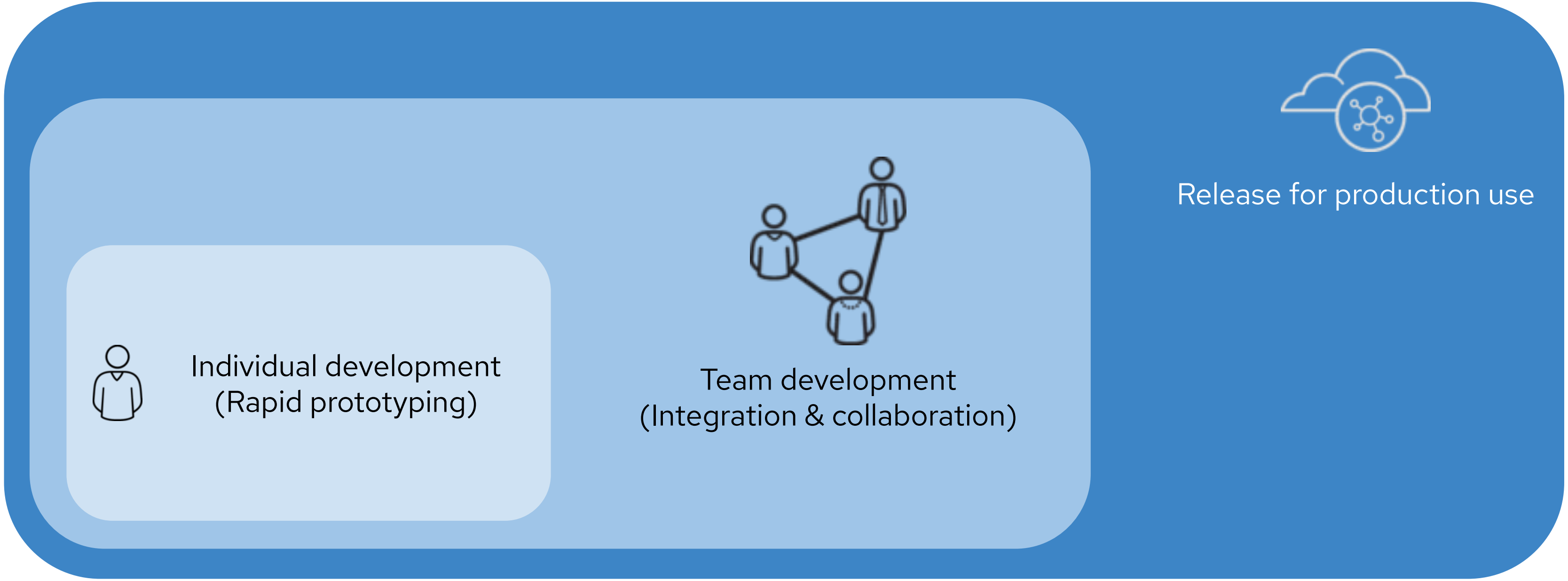
OpenShift DO - ODO

- Developer - Not Kubernetes experts
- ODO - a ‘git push’ like approach to creating and manipulating OCP applications in development



```
odo create nodejs node-app-rest
Validation
 ✓  Validating component [67ms]

odo url create node-app-rest
 ✓  URL node-app-rest created for component: node-app-rest

odo push
Validation

Applying URL changes
 ✓  URL node-app-rest:
http://node-app-rest-app-myrestapp.example.opentlc.com created

Pushing to component node-app-rest of type local
 ✓  Checking files for pushing [1ms]
 ✓  Waiting for component to start [1m]
 ✓  Syncing files to the component [360ms]
 ✓  Building component [16s]
 ✓  Changes successfully pushed to component

odo watch
```

OpenShift - Rapid innovation and collaboration

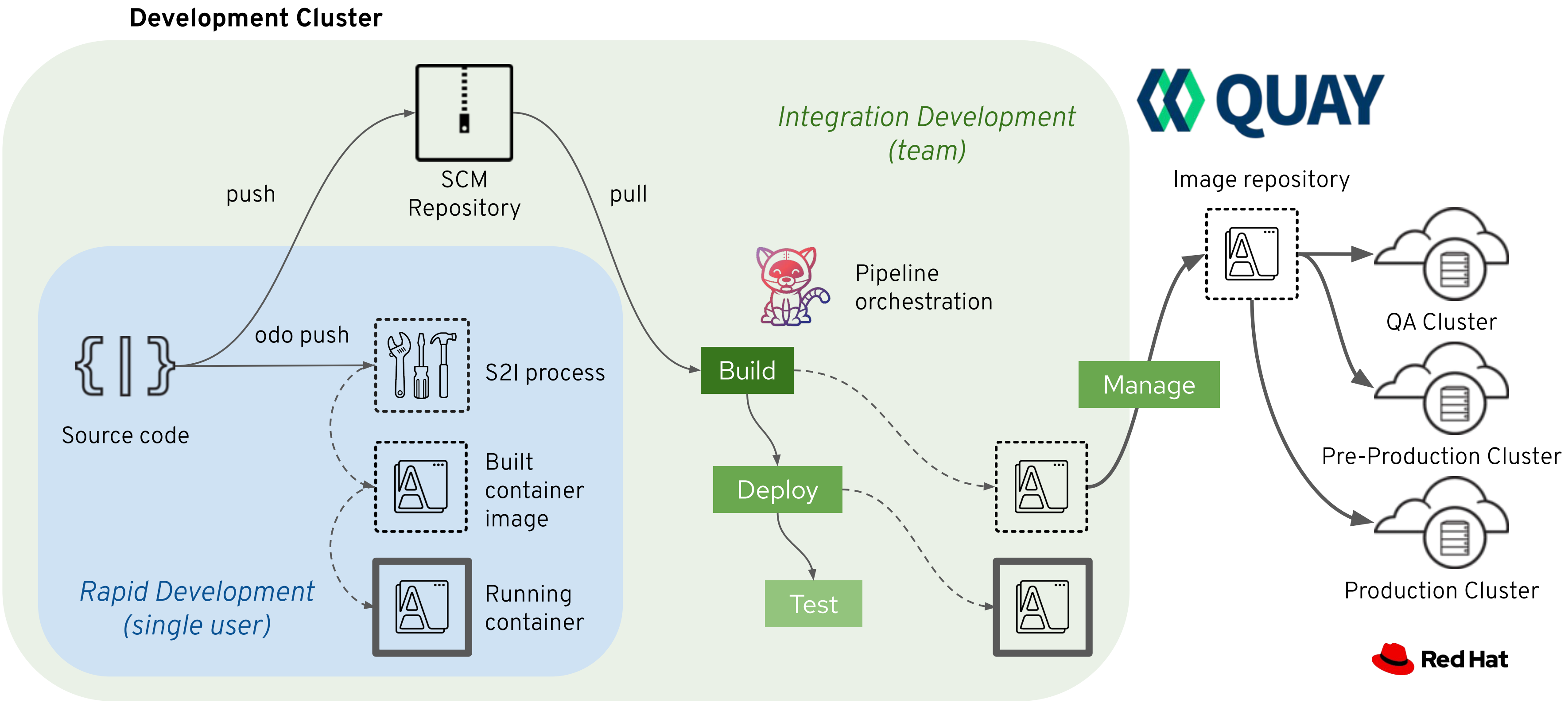


Individual development
(Rapid prototyping)

Team development
(Integration & collaboration)

Release for production use

The Development Process

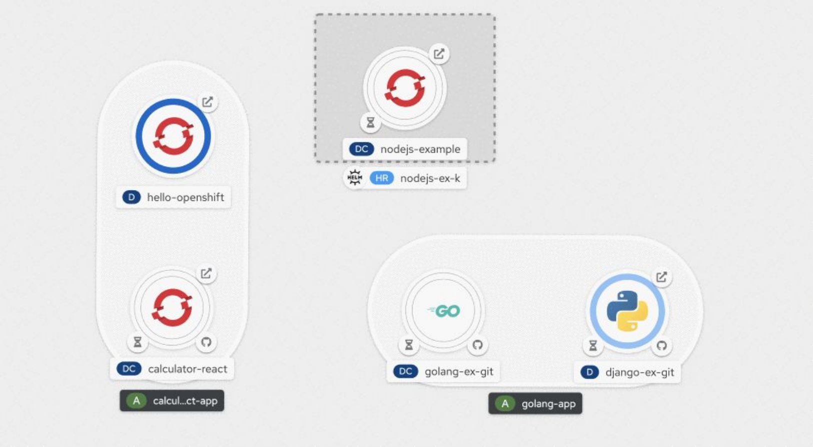


4 Application Interaction

Understanding Complex Applications

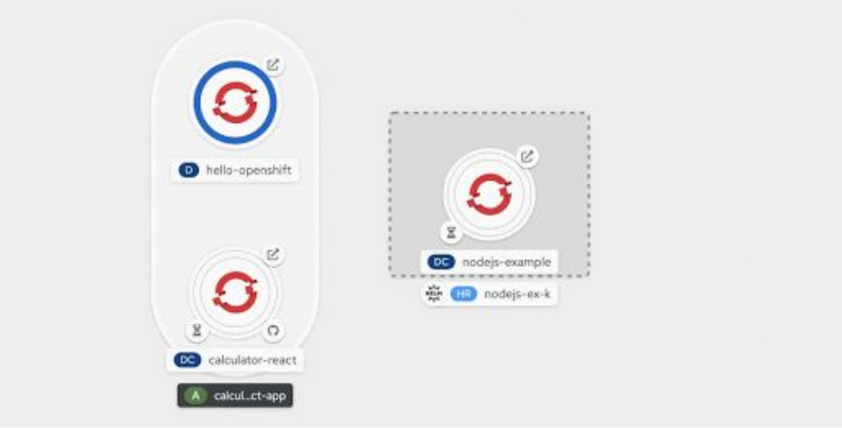
Add via popup action

- Developer stays focused and in context of app
- Streamline to access to “Add” features
- Maintains application grouping



Add via connector drop

- Developer stays focused and in context of app
- Easy access to “Add” feature and adds connector, either service binding or just visual connector



5

OpenShift comes to your laptop with CodeReady Containers

OpenShift on your laptop

CodeReady Containers

Provides a pre-built development environment based on Red Hat Enterprise Linux and OpenShift for quick container-based application development. Use with OpenShift on-premises or cloud.

Available for:

- Linux (KVM)
- Windows (Hyper-V)
- MacOS (hyperkit)

OpenShift 4.x: CodeReady Containers

- Linux, Windows and Mac
- Toolbar widget for quick access
- Simplified RHEL entitlement

6

A Kubernetes native Java framework: Quarkus

Quarkus - Kubernetes Native Java

Monolith

Cloud Native

Microservices

Serverless

Event-Driven Architecture

kubernetes

Istio

Knative

Benefit No. 1: Developer Joy

A cohesive platform for optimized developer joy:

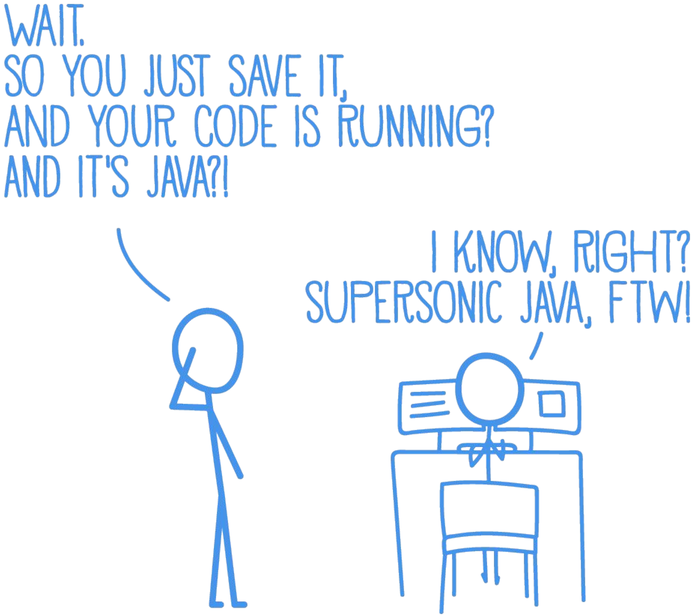
Zero config, live reload in the blink of an eye

Based on standards, but not limited

Unified configuration

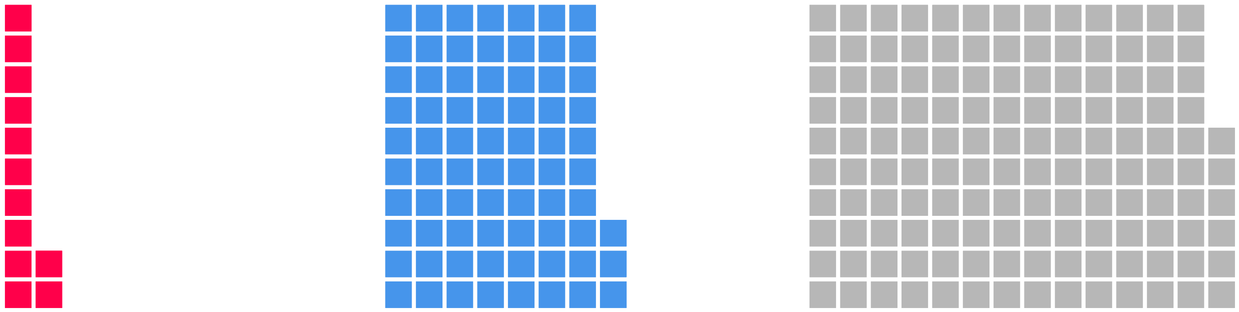
Streamlined code for the 80% common usages, flexible for the 20%

No hassle native executable generation



Benefit No. 2: Supersonic Subatomic Java

REST*



Quarkus + Native
(via GraalVM)
12 MB

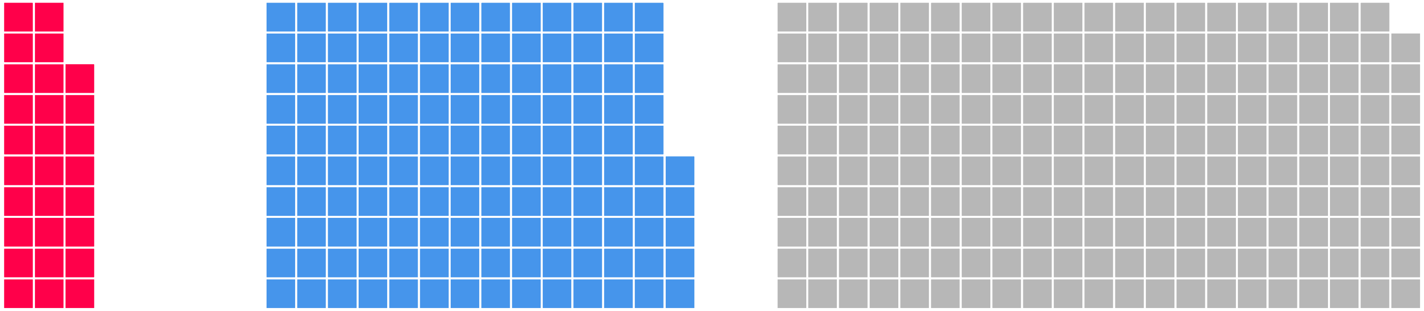
Quarkus + JVM
(via OpenJDK)
73 MB

Traditional
Cloud-Native Stack
136 MB

*Memory (RSS) in Megabytes, tested on a single-core machine

Benefit No. 2: Supersonic Subatomic Java

REST + CRUD*



Quarkus + Native
(via GraalVM)
28 MB

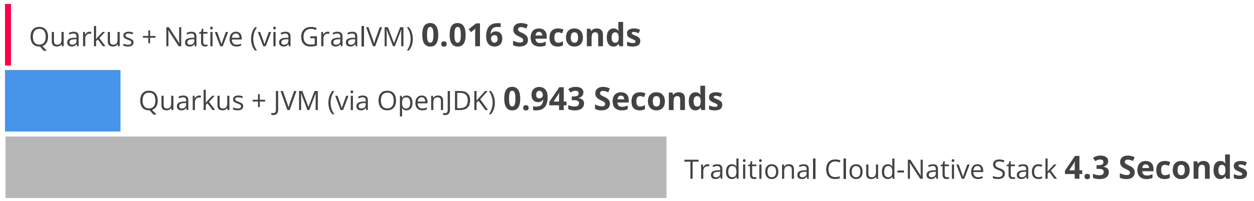
Quarkus + JVM
(via OpenJDK)
145 MB

Traditional
Cloud-Native Stack
209 MB

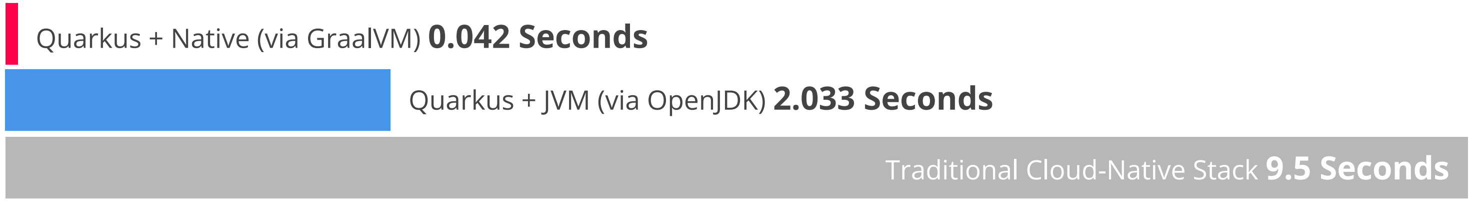
*Memory (RSS) in Megabytes, tested on a single-core machine

Benefit No. 2: Supersonic Subatomic Java

REST

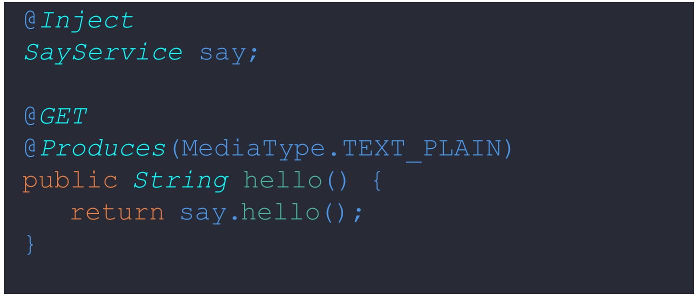


REST + CRUD

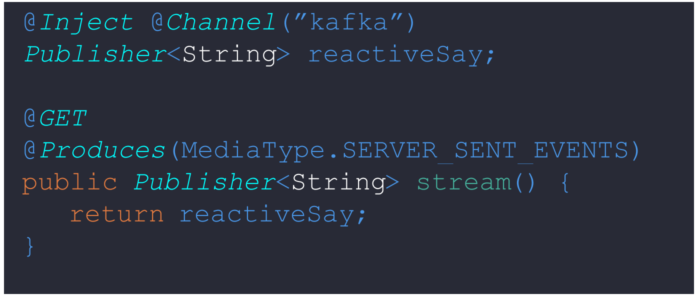


Time to first response

Benefit No. 3: Unifies Imperative and Reactive



```
@Inject
SayService say;

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



```
@Inject @Channel(”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
- 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

Quarkus provides a cohesive, fun to use, full-stack framework by leveraging a growing list of over fifty best-of-breed libraries that you love and use. All wired on a standard backbone.

Not a fan of reading? Hands-on learning.

…same…

- Free 2-week OpenShift 4 Cluster in the Cloud.
- Red Hat Developer Tutorial: Service Mesh.
- Red Hat Developer Content: CI/CD with Tekton (OpenShift Pipelines), Blog Series with Example.
- Serverless (kNative): Hands-on , or with Spring Boot.
- Camel-K connectors: Tutorial.
- Spring Boot too slow? Try Quarkus (container native Java framework).
- OpenShift Ireland User Group

Thank You

in linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

f facebook.com/redhatinc

twitter.com/RedHat