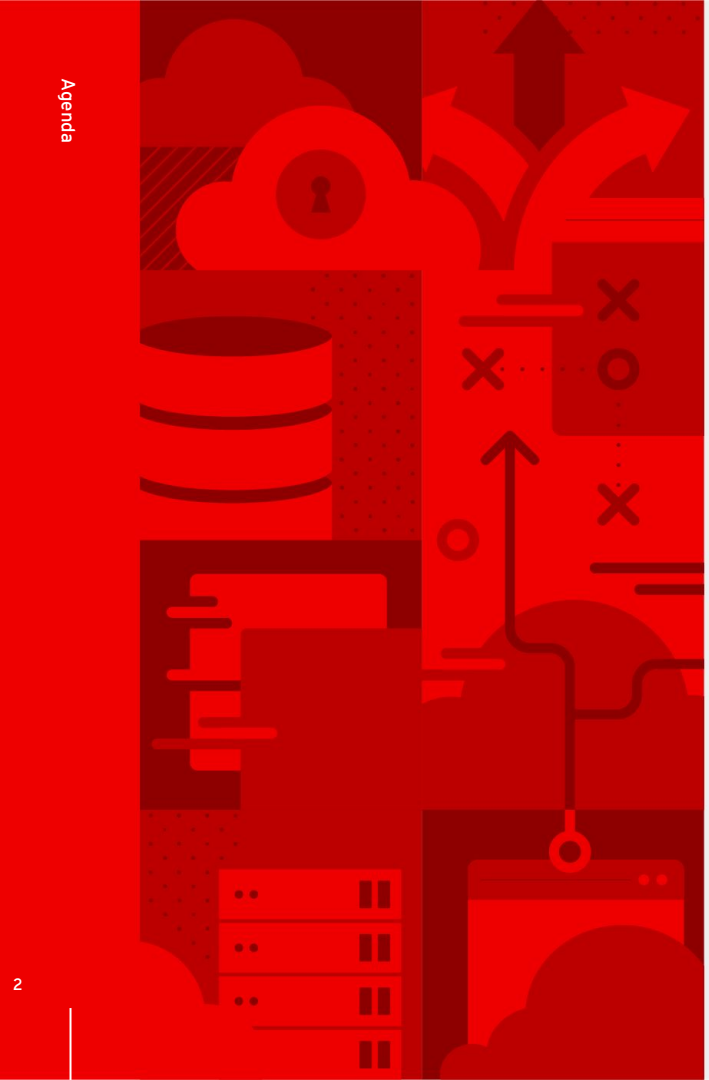
Red Hat Middleware

Red Hat Tech Exchange: Cloud Native Product/Dev Experience

Naina Singh
Senior Product Manager

Charles Moulliard
Software Engineering Manager

Red Hat



Agenda

RH MW : Product/Dev Experience

Halkyon

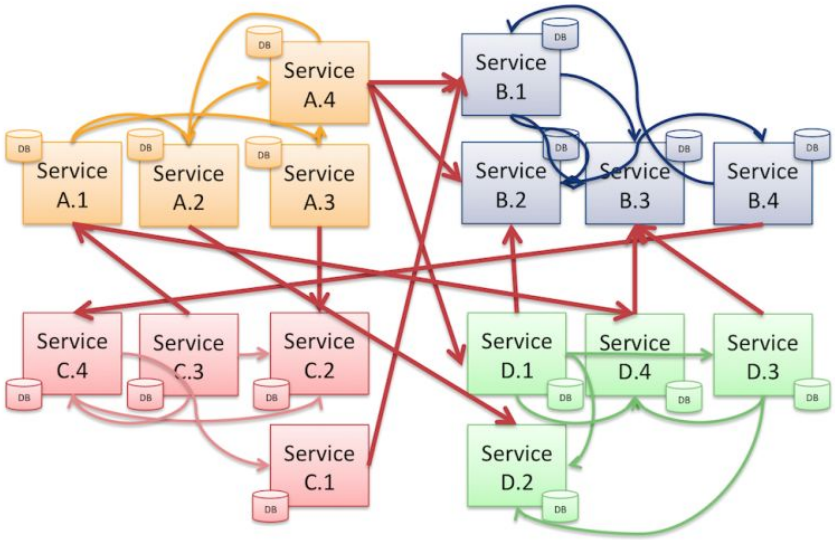
Red Hat

- What is the Cloud Native Product/Dev Experience about ?

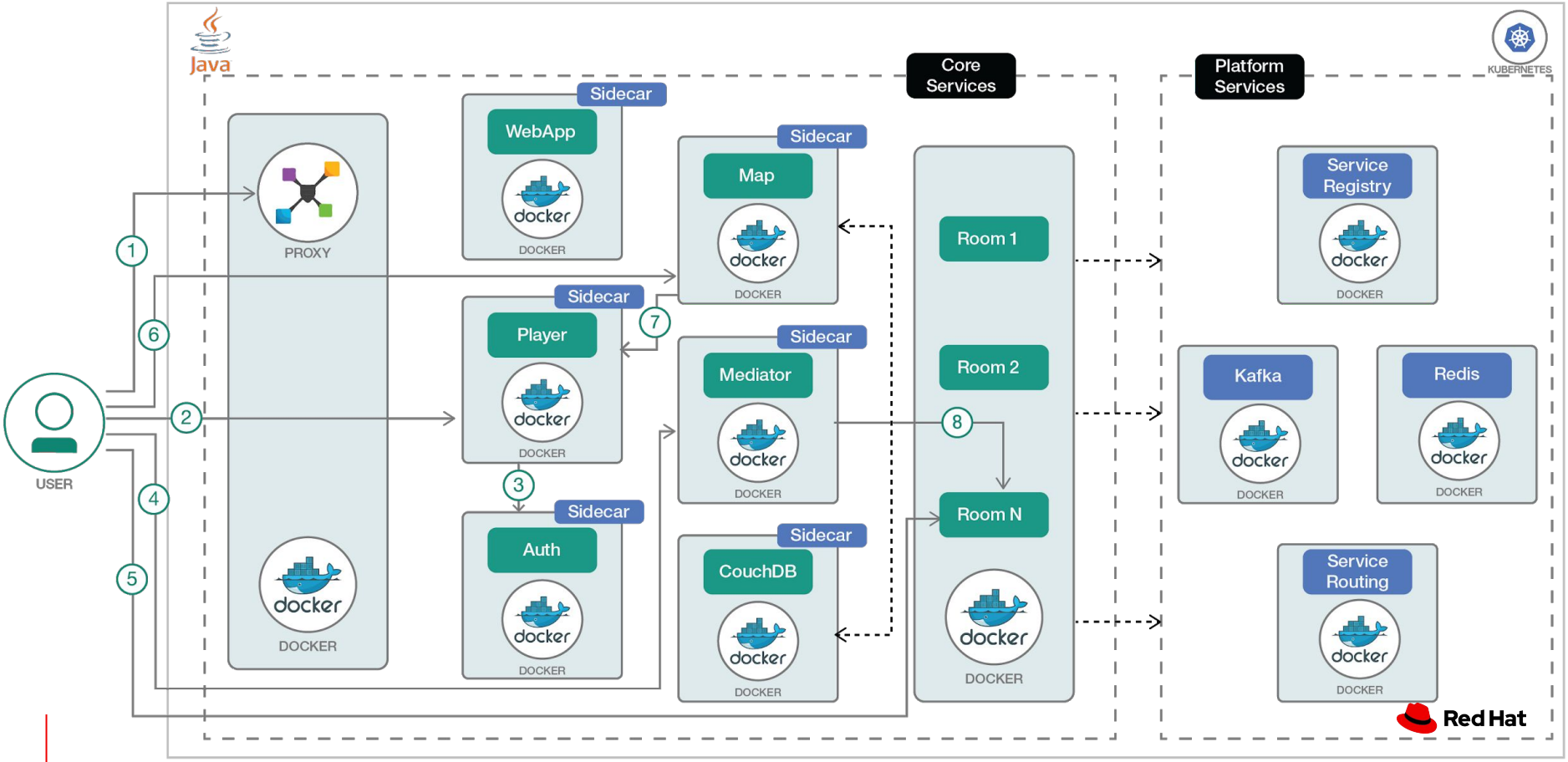
  “*Simplification of the process to package/deploy microservices on a k8s/ocp cluster based on 12 Factor guidelines*”

Microservices everywhere

- Design of a modern application embraces the microservice paradigm, supports multi-languages (java, nodejs, python, ...) & results in composite solutions where runtimes communicate over the network



Containers and k8 Orchestration



Consequences

- K8s resource/application
  - (Dockerfile) + DeploymentConfig + ImageStream + BuildConfig + Service + Route → 6 yaml files
  - ConfigMap/Secret + ServiceAccount + PV/PVC → 9 files
- Simple Use case of 2-3 microservices: 15-22 (= 2 x (6-9) + 1 x (3-5))
- Complex use case: hundred of resources to “qualify/test”

Consequence

- K8s resource/application
  - (Dockerfile) + DeploymentConfig + ImageStream + BuildConfig + Service + Route → **6 yaml files**
  - ConfigMap/Secret + ServiceAccount + PV/PVC → **9 files**
- Simple Use case of 2-3 microservices: **15-22** (= 2 x (6-9) + 1 x (3-5))
- **Complex use case: hundred** of resources to **“qualify/test”**

Customer - Return of Experience

*"The **complexity** of K8s - you have to **understand** & **create all the** objects (pods, deployment, service, route, ...). For most apps a **simpler deployment model** would be great"*

*"If you want to do **anything different** from the defaults then it seems very **complex**."*

“Local development with a multi component/service integration

*It is **almost impossible** to use OpenShift for **quick development iteration**. Can only be used for final testing once it is working in local environment.”*

From :OpenShift survey -2018

How can we help you as Developer ?

- K8s is great but we have to design/play with **xxxx** yaml files :-(
- Dekorate
  - Java **Utility** project generating **Manifests**
    - Kubernetes/OpenShift
    - HalkyonComponent
    - ServiceCatalog
  - Using Java **Annotations** or Annotation less approach

- Great but what about **12-Factor**, exposing services/routes, deploy a Database
- Halkyon
  - Aims to **simplify** the process of **deploying/linking** microservices on k8s
  - Configure **capabilities** such as Database, …
  - **Compatible** with K8s cluster or ocp
  - Multiple Stack Orchestration

- Need a **Tool** to play with technology, resolving cloud pattern
- Red Hat Developer Launcher
  - Generate Zip project or Git repo to built/deploy Application on OpenShift (Template)
  - Create/import applications
  - Use a collection of Examples / runtime / version
  - Deploys on OpenShift Cluster

- Do you have a Dev Cli Tool able to save my life ?
- hal CLI
  - A CLI tool to easily **create/scaffold** micro-services applications using the halkyon operator
  - Relies on Dekorate yaml manifests
  - **Push** source or binaries → pod
  - **Link/Bind** microservices
  - Populate **capabilities**

Solution: Adopt a Descriptive Model

- **What does it achieve?**
  - **Describe** what a runtime | service | link to be deployed/configured **is**
  - Represent an **intermediate**, more **readable** concept
  - Simplifies & reduces cognitive load on developers, helping them to be more productive

Solution: Use K8s Custom Resource

- **Why** ?
  - **Connect** the infrastructure, such as deployments/builds, to a **unique** object
  - **Operate** on semantically-related resources “atomically” via CR

Solution: Use K8s CRD/Operator

- **Why** ?
  - Move **complex** logic to an **operator** instead of replicating it between different tools (fmp, oc, odo, launcher, …)
  - **Resource** is described using a **versioned, documented** API = Spec
  - **CRUD** operations supported by default

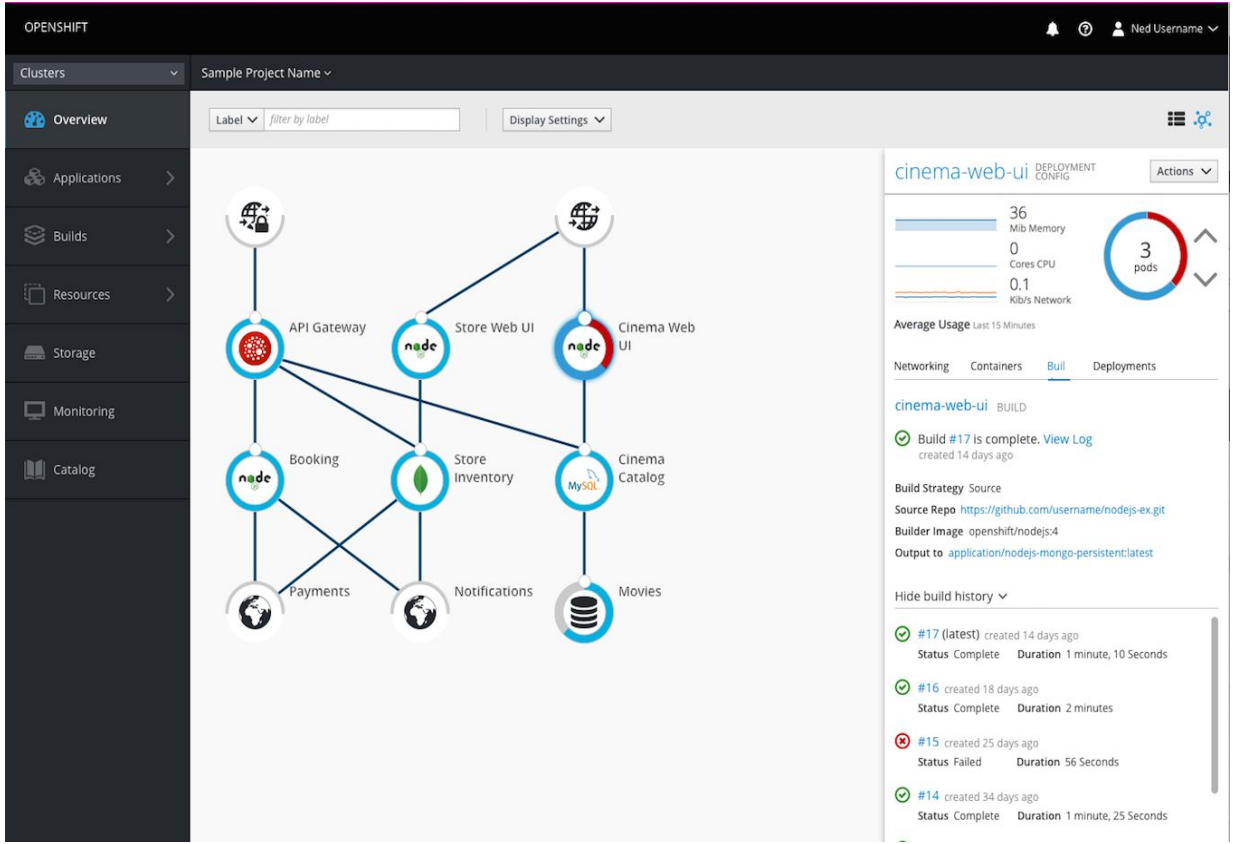
Solution: Adopt a **K8s Custom Resource**

- **Benefits !**
  - Adopt **Common Model Definition** : tools, UI, ...
  - Reduce **maintenance** costs of existing tools as they don’t have to **manage/generate** all the **k8s** resources

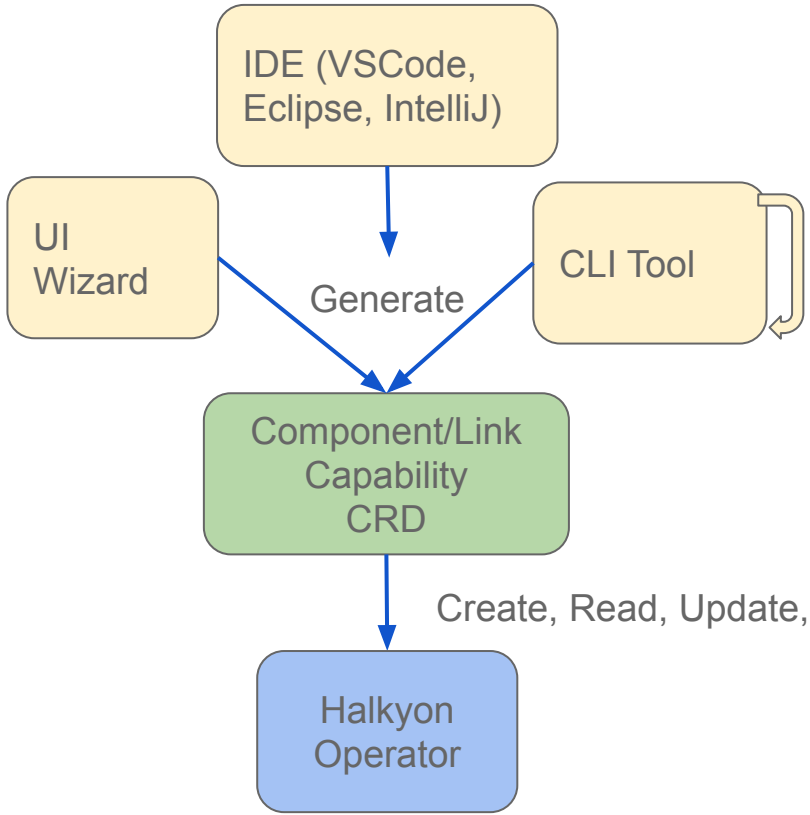
Solution: Adopt a **K8s Custom Resource**

- **Benefits !**
  - Improve Tooling → **Composition IDE Tool**
  - **Visually** display the components/services, their relations (UI, Terminal), status, … (management/monitoring)

Example: Visual Composition



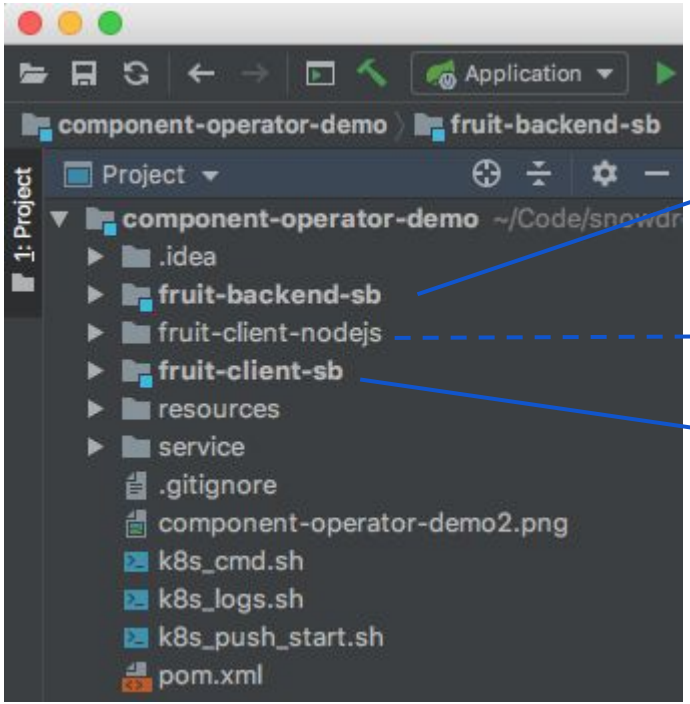
What about the Architecture ;-)



Iterative steps executing : hal
create myapp,
hal create sb1
hal create sb2
hal create xxx
hal link sb1 sb2,
hal service create
db-postgresql
...

How to compose an App

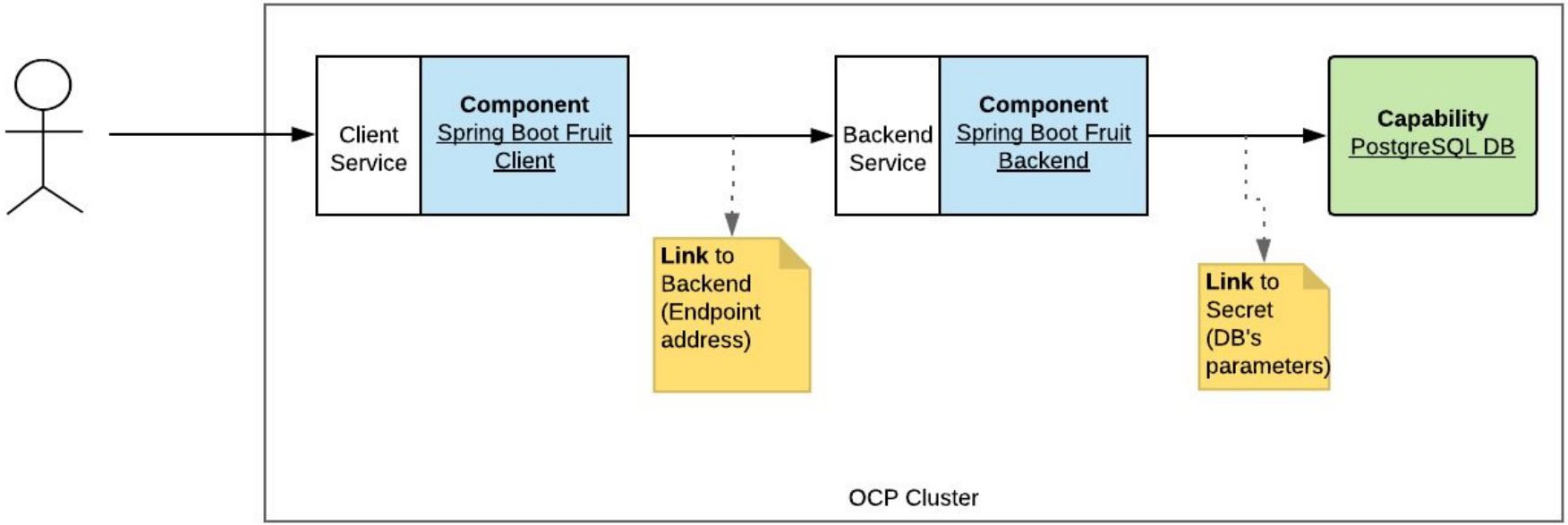
Java maven project -> modules

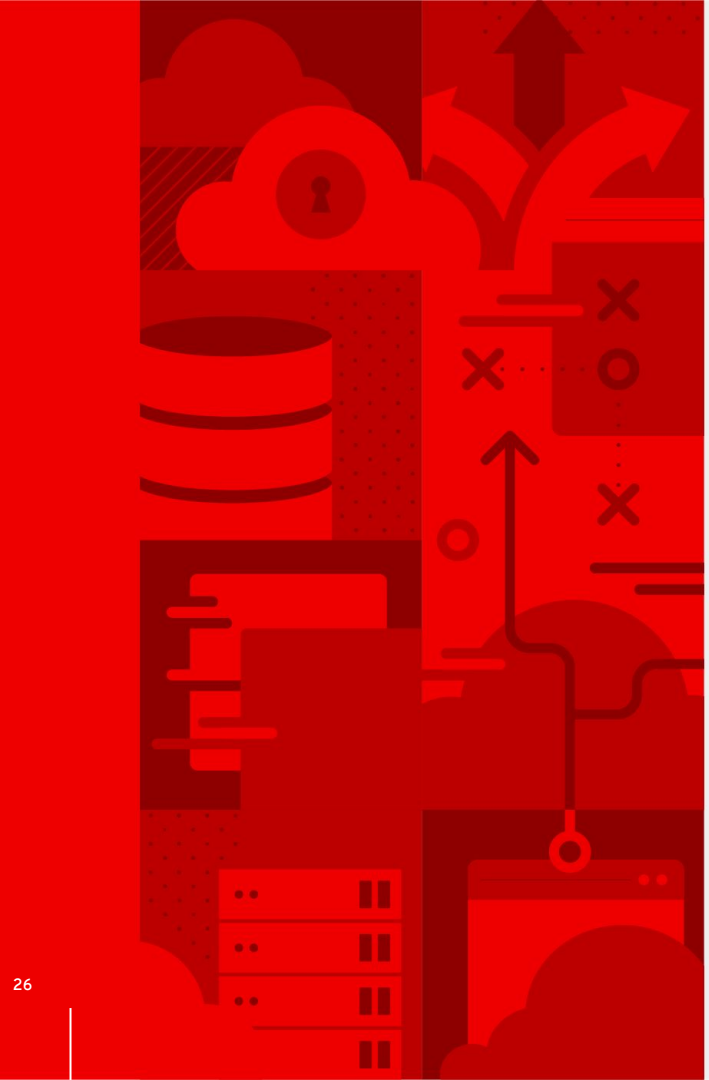


Component **B** - Spring Boot

Component x - Nodejs

Component **A** - Spring Boot


Client Service
Component
Spring Boot Fruit Client
Backend Service
Component
Spring Boot Fruit Backend
Capability
PostgreSQL DB
Link to Backend (Endpoint address)
Link to Secret (DB's parameters)
OCP Cluster



Questions & Next Steps

Red Hat