Running an OpenShift Environment on Your Workstation

June 2018 | Leon Levy – Solution Architect
<llevy@kovarus.com>

KOVARUS
Integrated IT Expertise

AGENDA

- OpenShift Overview
- MiniShift Overview and Install
- Logging Into OpenShift
- Deploy and Manage a Simple Application
- Java Microservice Application Demo
- Additional Topics for Next Time

KOVARUS
Integrated IT Expertise

OpenShift Container Platform

"vCenter for Containers?"

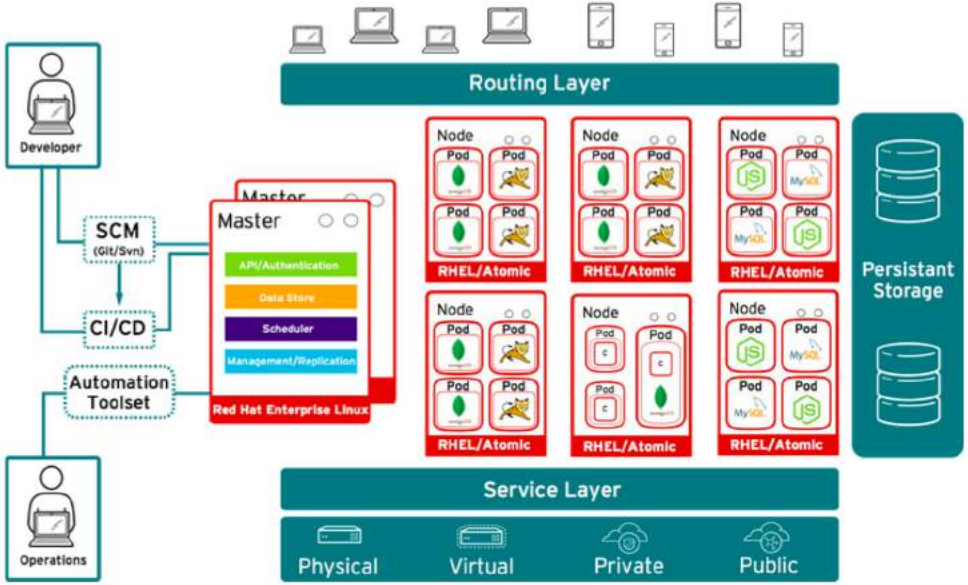
Unite developers and IT operations on a single platform to build, deploy, and scale applications

3

OpenShift at a glance

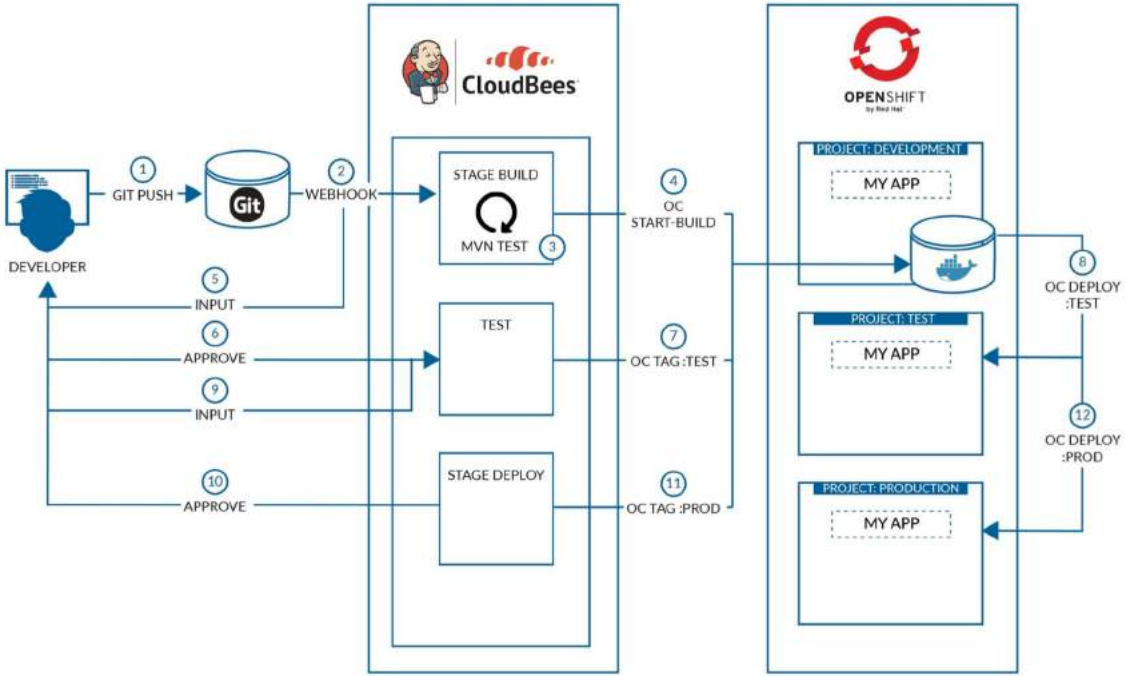
- PaaS
- Deploy application container from source S2I (Pod)
- Scale application pods for resiliency and load balancing
- Roll out code changes to different environments test/staging/production/etc…
  - On-prem/Off-prem
  - Virtual or physical servers
- CI/CD pipeline integration

OpenShift Environment Architecture 101



KOVARUS
Integrated IT Expertise

CI/CD Pipeline

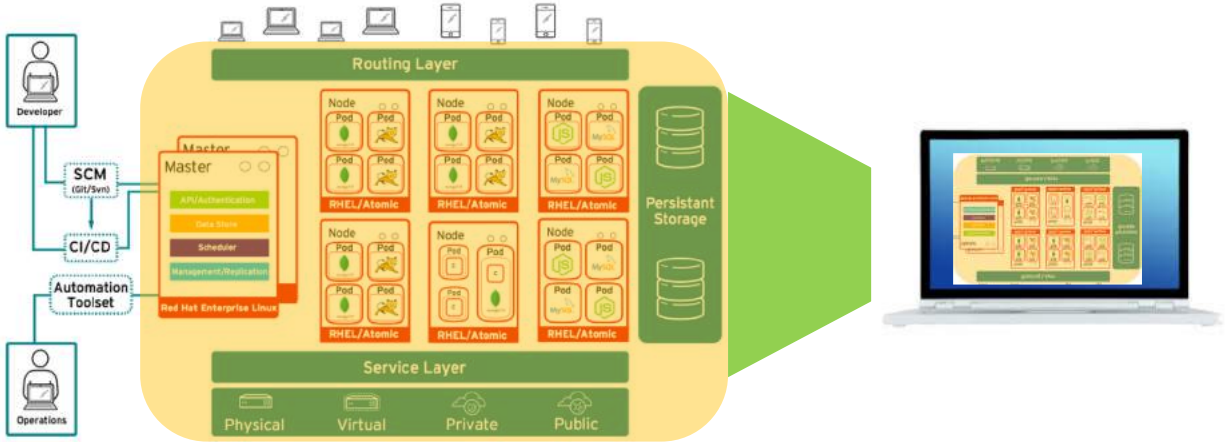


MiniShift Overview

Installation and Application Deployment

Minishift is a tool that helps you run OpenShift locally by running a single-node OpenShift cluster inside a VM. You can try out OpenShift or develop with it, day-to-day, on your local host.

MiniShift Environment



- Runs in a single VM
- Part of Red Hat Development Suite (CDK)
- Not Intended for production
- Only accessible from workstation

- There are other ways to deploy single service OpenShift instance

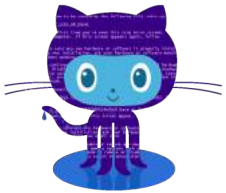
Lab Notes and Links

- Google Doc
  - http://goo.gl/m5EiAY

MiniShift Prerequisites

- Red Hat Developer Account (Free) 

  - https://developers.redhat.com
- Git client and Github account (if you want to modify code examples)
  - https://git-scm.com
  - https://github.com



- IDE (PyCharm CE/Atom/Notepad++/vi/etc..)
- Basic git knowledge
  - git clone/add/commit/push


Windows Prerequisites

- virtualbox

- Tools I like that make my life better on Windows
  - Chocolatey package manager - https://chocolatey.org
  - Cygwin (included with RH Dev Suite)
  - ConEmu
  - For local builds
    - jdk and maven installed
    - installing maven via choco will automatically install jdk
    - python
    - git

Download and install Red Hat Development Suite

- https://developers.redhat.com
  - Products -> Developer Tools -> Red Hat Development Suite -> Download (https://developers.redhat.com/products/devsuite/download)
    - Download version 2.2 and not the latest

- Includes
  - MiniShift (MiniShift RHEL VM and startup scripts)
  - OC (OpenShift CLI)
  - VirtualBox (latest supported version)
  - Red Hat JBOSS Developer Studio (IDE)
  - cygwin
  - And more….!!!!

Install location: *C:\Program Files\DevelopmentSuite (Windows)*
/Applications/ DevelopmentSuite (Mac)

KOVARUS
Integrated IT Expertise

Start Minishift

- Via command line
  - `minishift start`

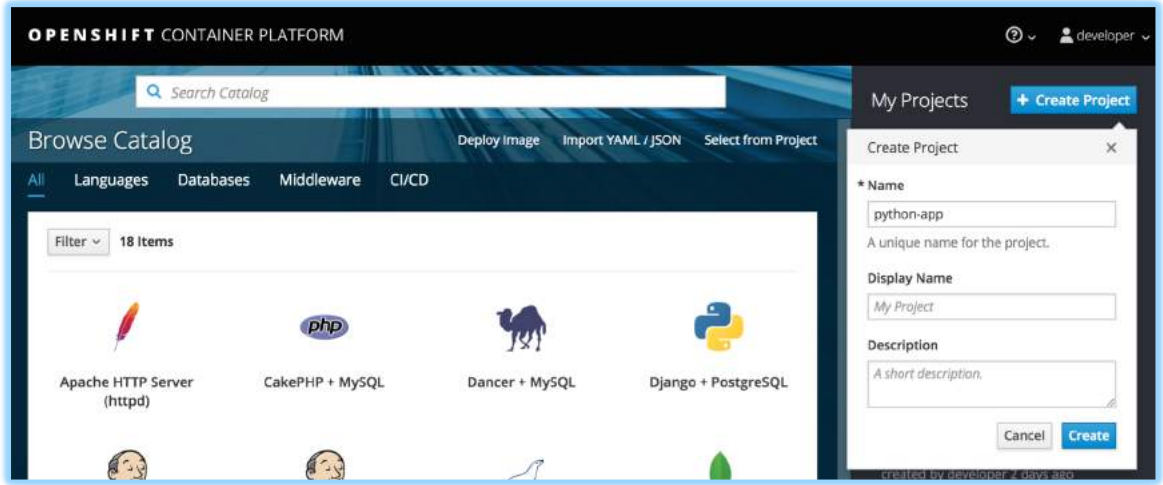
```
➜ ~ minishift start
\-- Starting profile 'minishift'
-- Checking if requested hypervisor 'virtualbox' is supported on this platform ... OK
-- Checking if VirtualBox is installed ... OK
-- Checking the ISO URL ... OK
-- Starting local OpenShift cluster using 'virtualbox' hypervisor ...
-- Starting Minishift VM ..... OK
-- Registering machine using subscription-manager
   Red Hat Developers or Red Hat Subscription Management (RHSM) username: leonlevy
   Red Hat Developers or Red Hat Subscription Management (RHSM) password: [HIDDEN]
   Registration in progress ....... OK [14s]
-- Checking for IP address ... OK
-- Checking if external host is reachable from the Minishift VM ...
   Pinging 8.8.8.8 ... OK
-- Checking HTTP connectivity from the VM ...
   Retrieving http://minishift.io/index.html ... OK
-- Checking if persistent storage volume is mounted ... OK
-- Checking available disk space ... 61% used OK
-- OpenShift cluster will be configured with ...
   Version: v3.7.14
-- Checking `oc` support for startup flags ...
   host-config-dir ... OK
   host-data-dir ... OK
   host-pv-dir ... OK
   host-volumes-dir ... OK
   routing-suffix ... OK
Starting OpenShift using registry.access.redhat.com/openshift3/ose:v3.7.14 ...
OpenShift server started.

The server is accessible via web console at:
    https://192.168.99.100:8443
```

Login to MiniShift CLI and WebUI

- CLI
  - `oc login https://<ip address from startup>:8443`
- WebUI
  - Point browser to `https://<ip address from startup>:8443`
- Username/Password – developer/dev
  - `minishift ip in case you forget IP address`
    - `oc login https://$(minishift ip):8443`

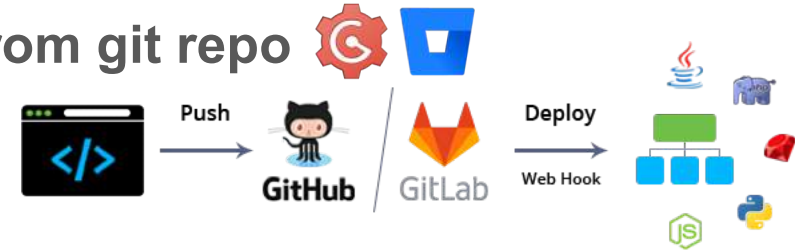
Create a new project

- CLI
  - `oc new-project python-app`
- WebUI



Create new application in project

- More than one way to do this
  - Direct from local build (maven fabric8 module in java)
  - Deploy a prebuilt container
  - **S2I from git repo**

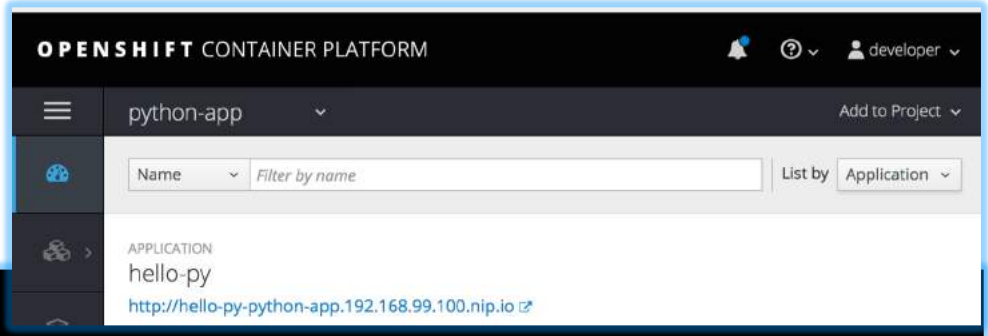


Sample Python/Flask Application

https://github.com/kovarus/os-sample-python-flask

Deploy Python application S2I

- `oc new-app https://github.com/kovarus/os-sample-python-flask.git --name hello-py`
- `oc logs -f bc/hello-py`
- `oc expose service hello-py`
- `oc get routes`




```
Pushed 6/6 layers, 100% complete
Push successful
➜  ~ oc expose svc hello-py
route "hello-py" exposed
➜  ~ oc get routes
NAME       HOST/PORT                                PATH      SERVICES   PORT       TERMINATION   WILDCARD
hello-py   hello-py-python-app.192.168.99.100.nip.io           hello-py   8080-tcp                 None
```

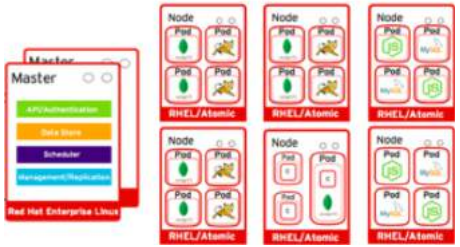
Python App


```
15
16  @application.route('/')
17  @application.route('/index')
18  def index():
19      now = datetime.datetime.now()
20      user = {'username': 'User'}
21      hostname = socket.gethostname()
22      return render_template('index.html', title='Flask on OpenShift',
23                             user=user,
24                             hostname=hostname,
25                             time=now.strftime('%m-%d-%Y %H:%M'))
26
```

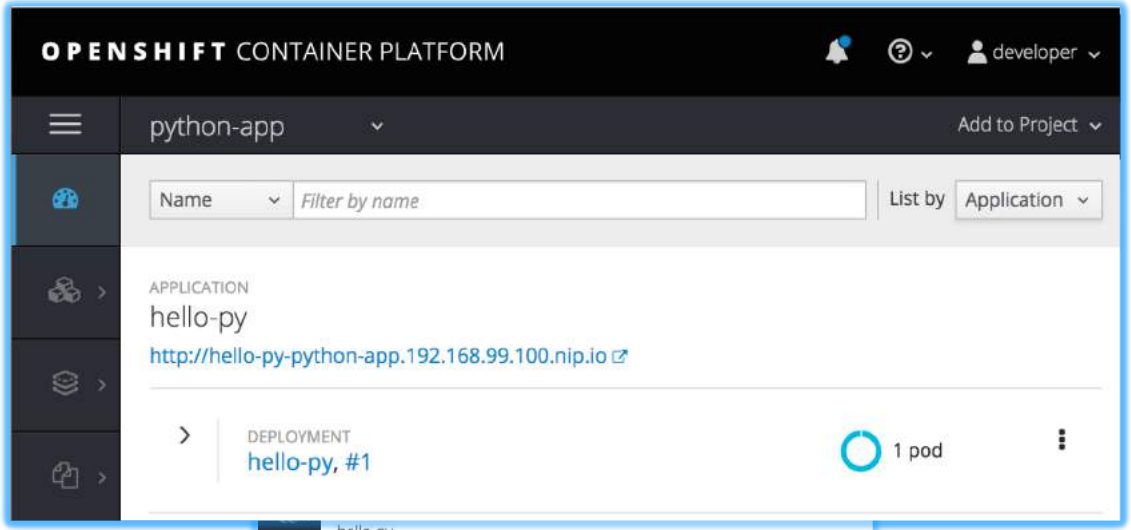
12 lines (11 sloc) | 259 Bytes

```
1  <html>
2      <head>
3          <title>{{ title }} - Example</title>
4      </head>
5      <body>
6          <h1>Hello, {{ user.username }}!</h1>
7          <p>Running in host/container: {{ hostname }}</p>
8          <br>
9          <p>It is now: {{ time }}.</p>
10     </body>
11 </html>
```

Scaling application




```
oc scale --replicas=3 dc hello-py
```



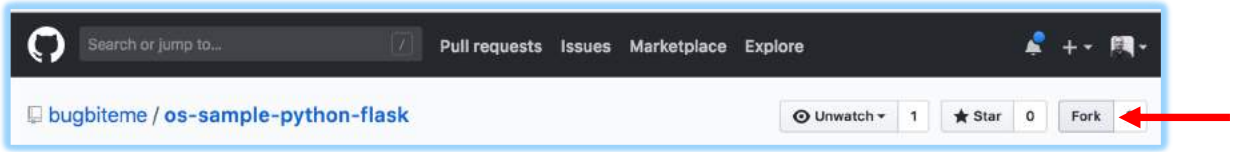
KOVARUS
Integrated IT Expertise

Deploy Build (i.e. code changes)

- `oc start-build hello-py`
- `oc logs -f bc/hello-py` **(track build progress)**

Code change example (using git) Slide 1 of 3

- Go to https://github.com/kovarus/os-sample-python-flask
- Fork project to your own account



- Deploy new project and app from forked repo

```
oc new-project python-app2
oc new-app https://github.com/<your repo>/os-sample-python-flask.git –name hello-py
oc expose service hello-py
oc get routes
```

- Make sure app deployed as expected (view via web browser)
- Scale to 3 instances
  - `oc scale --replicas=3 dc hello-py`

Code change example (using git) Slide 2 of 3

- Clone forked repo to local system
  - `git clone https://github.com/<your repo>/os-sample-python-flask.git`
- Change some code!
  - `cd os-sample-python-flask`
- Edit `wsgi.py` (vi, notepad++, PyCharm, Atom, etc…)

```
15
16  @application.route('/')
17  @application.route('/index')
18  def index():
19      now = datetime.datetime.now()
20      user = {'username': 'State of CA'}
21      hostname = socket.gethostname()
22      return render_template('index.html', title='Flask on OpenShift',
23                             user=user,
24                             hostname=hostname,
25                             time=now.strftime('%m-%d-%Y %H:%M'))
```

Code change example (using git) Slide 3 of 3

- Save code and check changes into github repo

```
git add *
git commit -m "doing some code changes”
git push
```

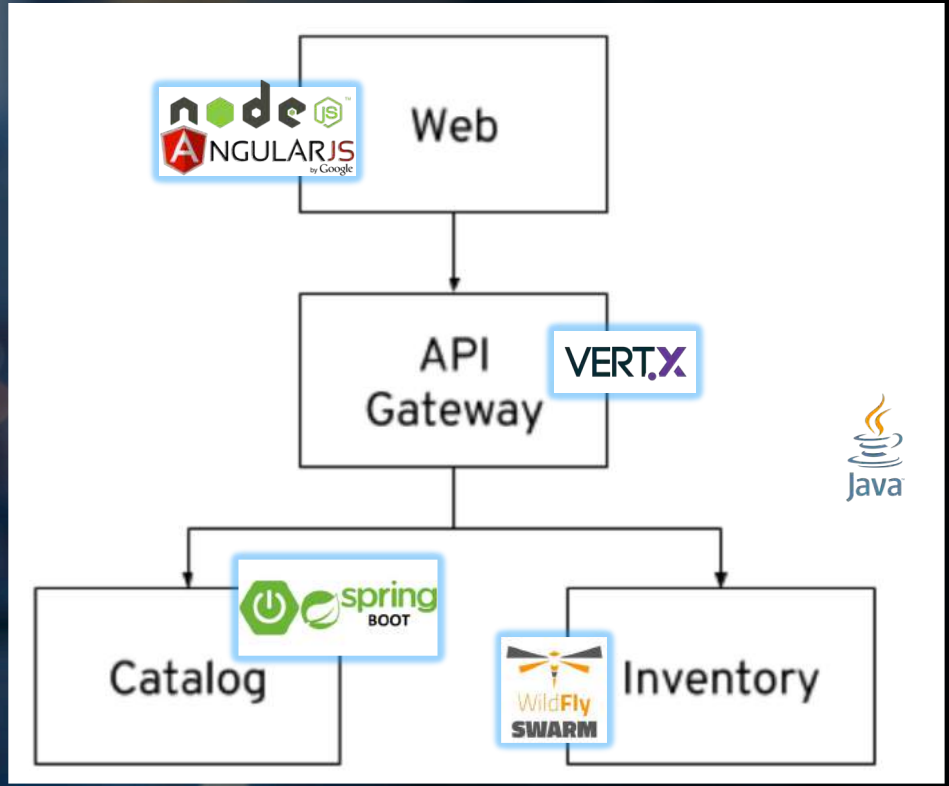
- Build and deploy code changes

```
oc start-build hello-py; oc logs -f bc/hello-py
```

- Refresh web-browser to view changes

Java Microservice Application

Demo



Components – Coolstore App (Kovarus Edition)

- **Inventory** - REST API for obtaining inventory from DB using Wildfly Swarm Java framework
  - http://wildfly-swarm.io
- **Catalog** - REST API for obtaining inventory from DB using Spring Boot Java framework
  - https://spring.io/projects/spring-boot
- **API Gateway** – Gateway for Inventory and Catalog using vert.x Java framework
  - https://vertx.io
- **Web** – Web frontend to end-user using NodeJS and AngularJS
  - https://nodejs.org
  - https://angular.io

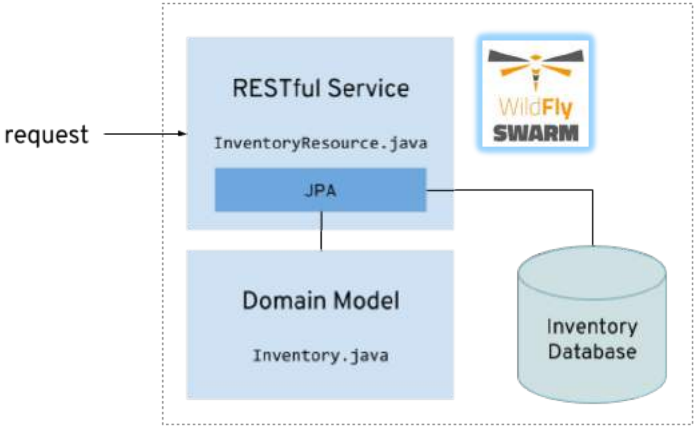
Easy to follow instructions to deploy it yourself!

Component	Github repo
Inventory	https://github.com/kovarus/inventory-wildfly-swarm
Catalog	https://github.com/kovarus/catalog-spring-boot
API Gateway	https://github.com/kovarus/gateway-vertx
Web	https://github.com/kovarus/web-nodejs

KOVARUS
Integrated IT Expertise

Inventory

- https://github.com/kovarus/inventory-wildfly-swarm

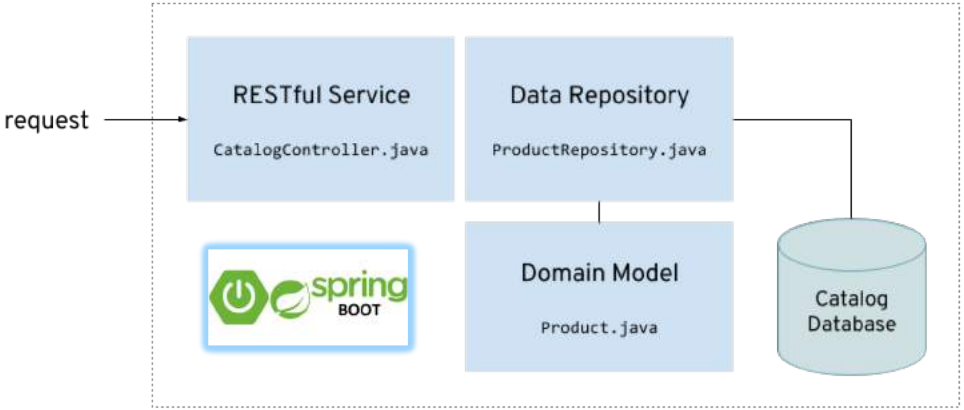


http://inventory-coolstore.192.168.99.100.nip.io/api/inventory/329299

```
{
        "itemId": "329299",
        "quantity": 35
}
```

Catalog

- https://github.com/kovarus/catalog-spring-boot



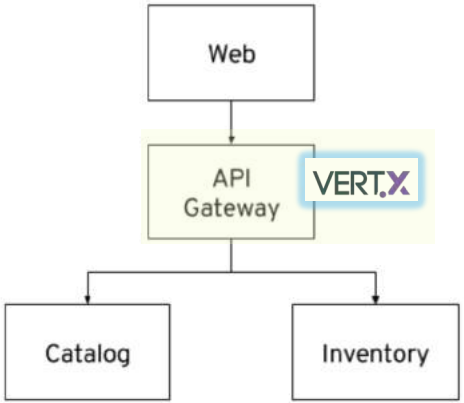
http://catalog-coolstore.192.168.99.100.nip.io/api/catalog

```
[
  {
      "itemId": "329299",
      "name": "Red Fedora",
      "description": "Official Red Hat Fedora",
      "price": 34.99
   },
   {
      "itemId": "329199",
      "name": "Forge Laptop Sticker",”…
```

KOVARUS Integrated IT Expertise

API Gateway

- https://github.com/kovarus/gateway-vertx



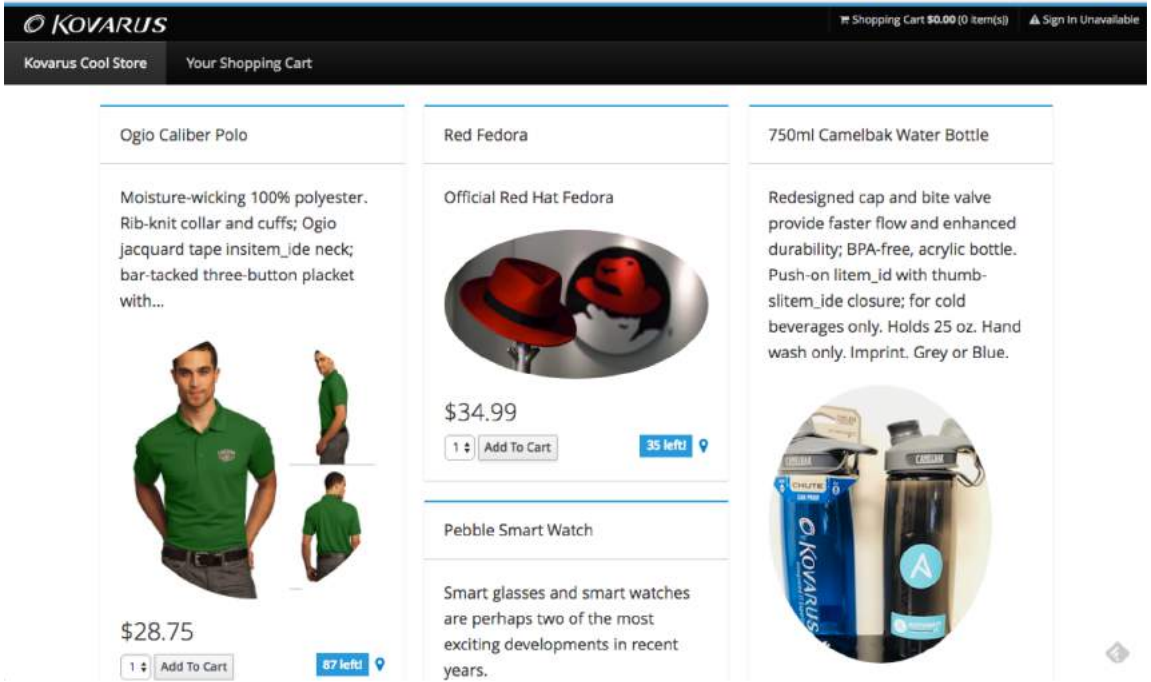
http://gateway-coolstore.192.168.99.100.nip.io/api/products


```
[
  {
    "itemId": "329299",
    "name": "Red Fedora",
    "description": "Official Red Hat Fedora",
    "price": 34.99,
    "availability": {
      "quantity": 35
    }
  },
  {
    "itemId": "165613",
    "name": "Solid Performance Polo",
    "description": "Moisture-wicking, antimicrobial 100% polyester d
    "price": 17.8,
    "availability": {
      "quantity": 45
    }
  },
  {
    "itemId": "165954",
    "name": "750ml Camelbak Water Bottle",
    "description": "Redesigned cap and bite valve provide faster flo
    closure; for cold beverages only. Holds 25 oz. Hand wash only. I
    "price": 6,
    "availability": {
      "quantity": 43
    }
```


KOVARUS
Integrated IT Expertise

Web frontend

- https://github.com/kovarus/web-nodejs
- http://web-coolstore.192.168.99.100.nip.io



For more information…

- OpenShift Container Platform 3.7 Documentation
  - https://docs.openshift.com/container-platform/3.7
- OpenShift Cloud-native Roadshow
  - http://guides-cdk-roadshow.b9ad.pro-us-east-1.openshiftapps.com
- Kovarus Github repos (includes all the code seen here)
  - https://github.com/kovarus
- Full-blown coolstore app source code and instructions
  - https://github.com/jamesfalkner/coolstore-microservice-brms

Advanced Topics

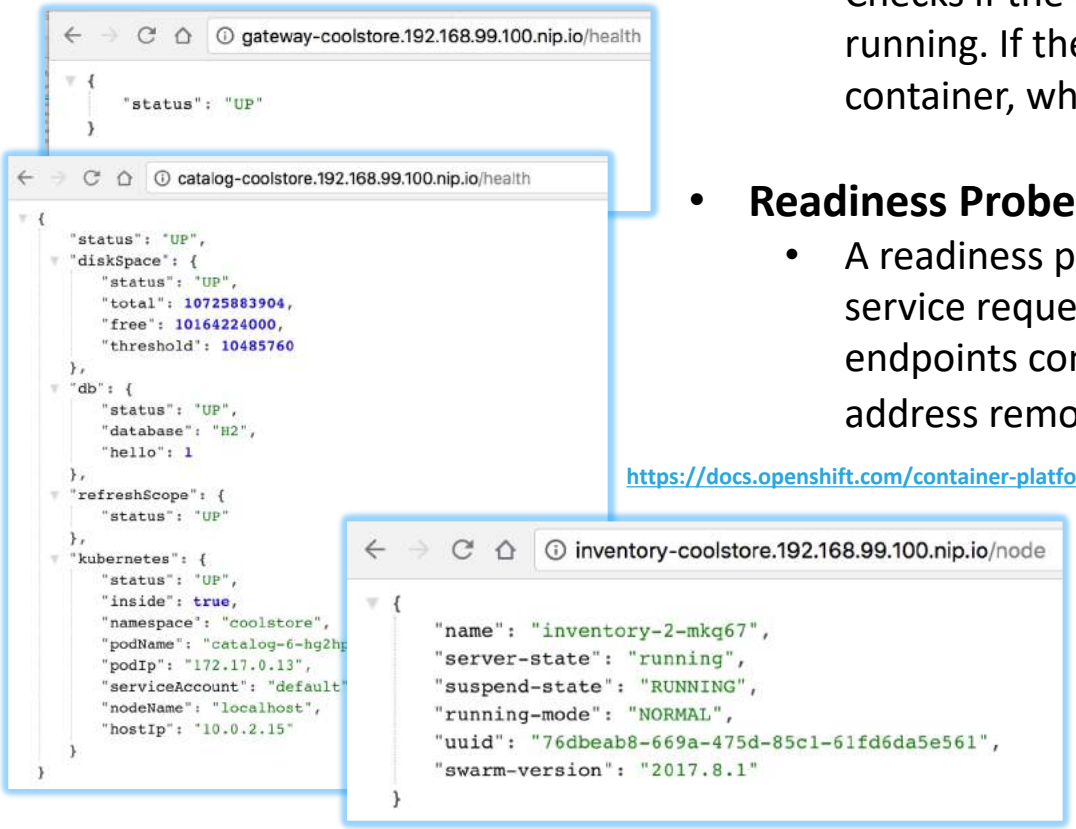
- Monitoring Application Health
- Service Resilience and Fault Tolerance
- Managing Application Configuration
- Automating Deployments Using Pipelines (future)
- Debugging Applications (future)
- Twelve Factor Applications

Monitoring Application Health – coolstore Example

- **Liveness Probe**
  - Checks if the container in which it is configured is still running. If the liveness probe fails, the kubelet kills the container, which will be subjected to its restart policy.

- **Readiness Probe**
  - A readiness probe determines if a container is ready to service requests. If the readiness probe fails a container, the endpoints controller ensures the container has its IP address removed from the endpoints of all services.

https://docs.openshift.com/container-platform/3.7/dev_guide/application_health.html#container-health-checks-using-probes



KOVARUS
Integrated IT Expertise

Service Resilience and Fault Tolerance

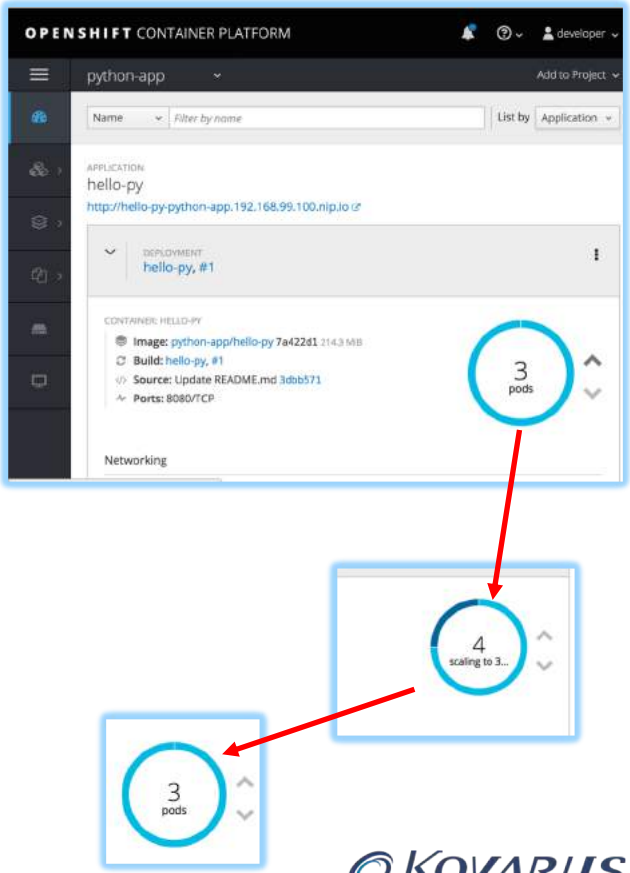
- Ensure you are back in the python-app2 project
  - `oc project python-app2`
- Ensure there are 3 containers/pods running
  - `oc scale --replicas=3 dc hello-py`
- Run command to view pods
  - `oc get pods`

```
➜ ~ oc get pods
NAME                READY     STATUS      RESTARTS   AGE
hello-py-12-jxs8f   1/1       Running     0          1h
hello-py-12-l8q6z   1/1       Running     0          42m
hello-py-12-ppk6z   1/1       Running     0          1h
```

- Delete one of the pods running
  - `oc delete pod hello-py-<build id>`

```
➜ ~ oc delete pod hello-py-12-jxs8f
pod "hello-py-12-jxs8f" deleted
```

- Observe self-healing (via Web GUI)



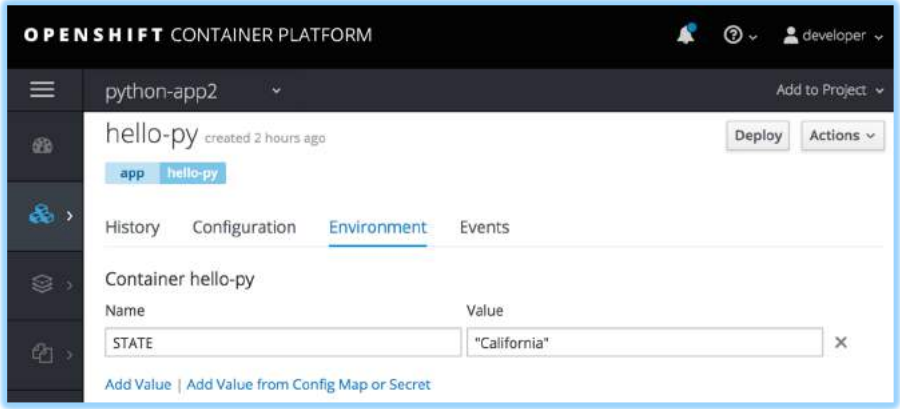
KOVARUS
Integrated IT Expertise

Managing Application Configuration

Set environment variables at time of application creation or post-deployment

```
$ oc new-app postgresql-persistent \
    --param=DATABASE_SERVICE_NAME=inventory-postgresql \
    --param=POSTGRESQL_DATABASE=inventory \
    --param=POSTGRESQL_USER=inventory \
    --param=POSTGRESQL_PASSWORD=inventory \
    --labels=app=inventory
```

GUI Interface

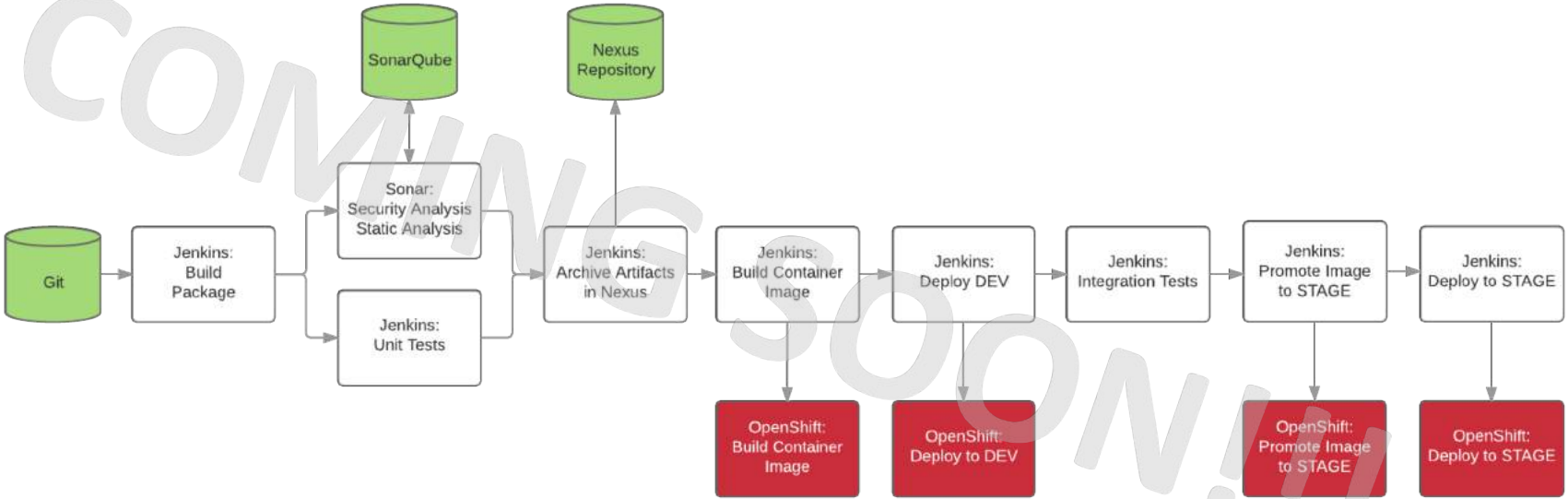


Using config map *.yml file

```
$ cat <<EOF > ./project-stages.yml
project:
  stage: prod
swarm:
  datasources:
    data-sources:
      InventoryDS:
        driver-name: postgresql
        connection-url: jdbc:postgresql://inventory-postgresql:5432/inventory
        user-name: inventory
        password: inventory
EOF
```

```
$ oc create configmap inventory --from-file=./project-stages.yml
```

Automating Deployments Using Pipelines



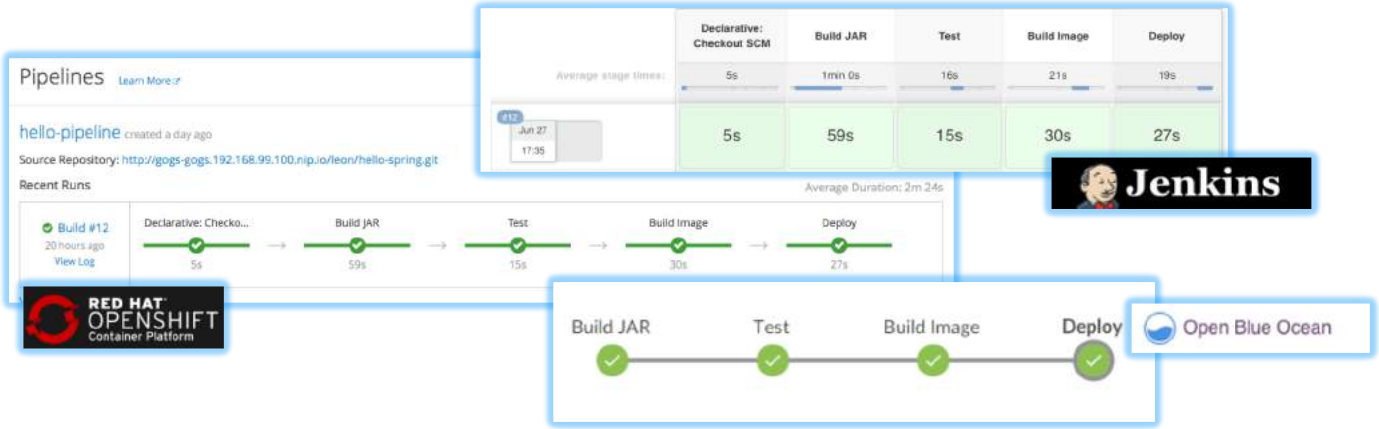
https://github.com/OpenShiftDemos/openshift-cd-demo

In the meantime:
https://github.com/kovarus/hello-spring

Simple pipeline example

https://github.com/kovarus/hello-spring

- Set up git repo inside OpenShift
- Clone public repo to local repo
- Deploy application from local git repo
- Set up pipeline (Jenkins)
- Webhook configuration for auto deploy after code push to git
- Unit testing



KOVARUS Integrated IT Expertise

Twelve Factor Applications - https://12factor.net

I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

IV. Backing services
Treat backing services as attached resources

V. Build, release, run
Strictly separate build and run stages

VI. Processes
Execute the app as one or more stateless processes

VII. Port binding
Export services via port binding

VIII. Concurrency
Scale out via the process model

IX. Disposability
Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity
Keep development, staging, and production as similar as possible

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin/management tasks as one-off processes

KOVARUS
Integrated IT Expertise