Automated Application Containerization

Journey to OpenShift with RHAMT

Zohaib Khan
App Modernization Lead
Red Hat - NAPS
zkhan@redhat.com
@zeebluejay

Marc Zottner
AMM Program Manager
Red Hat - EMEA
mzottner@redhat.com

2018/05/08
AGENDA

- Containerization: Why and What?
- Automating the approach
- Next steps
WHAT IS CONTAINERIZATION?
WHAT IS CONTAINERIZATION?

Packaging of a configured application and all its dependencies into a light, portable, cloud-ready sandbox.

- Physical host
- Virtual host
- Container
NEW POSSIBILITIES

SINGLE CONCERN

Container 1
Single Concern

Container 2
Single Concern

Deployment Unit (Pod)

HIGH OBSERVABILITY

Container

process health
readiness
liveness
metrics
tracing
logs

LIFECYCLE CONFORMANCE

Container

SIGTERM
SIGKILL
PreStop
PostStart

IMAGE IMMUTABILITY

Container

app.jar
java

Dev Test Prod

PROCESS DISPOSABILITY

Container

start/stop

SELF-CONTAINMENT

Container

app.jar
java

Build time
Run time

Storage

Configs
NEW WAYS TO DELIVER

SELF SERVICE
BLUE GREEN DEPLOYMENTS
CANARY RELEASES

A/B TESTING
END-TO-END AUTOMATION
ROLLING UPGRADES

#redhat #rhsummit
ECONOMICS

Change velocity vs. Number of hurdles eliminated

- Initial effort to see visible change
- Inflection point
- Accelerated value delivery

Resources, Time, Budgets

Maintain current portfolio

Innovation, differentiation, growth

19%

70%

#redhat #rhsummit
BEST KEPT SECRETS

1) **Not every** application fits into a container!

2) Containers are **not only** for elite-apps.

3) Conducted transformations must be driven by **business value**.
OPTIONS FOR CONTAINERIZATION

- **RETAI**N: Keep and don’t touch for now.
- **RETI**RE: Decommission end-of-life application.
- **REHOST** (Lift & Shift): Package existing application in a container with as few changes as possible.
- **REPLATFORM** (Lift & Reshape): Change the underlying platform (runtime, framework, middleware, operating system).
- **REFAC**TORY (Extend / Rewrite): Redesign code to take advantage of the new platform (extend, strangle, rewrite).
- **REPU**RCHASE (Drop & Shop): Replace by Commercial off-the-shelf (COTS) or Software-as-a-Service (SaaS).

Transformation Efforts / Complexity:
- Low
- High
FULL-AUTOMATED APPROACH
CONTAINERIZATION JOURNEY

1. Portfolio discovery and inventory
2. Readiness assessment
3. Automated Deployments
4. Import applications
5. Transform applications
6. Validate, promote, decommission
CONTAINERIZATION JOURNEY

1. Portfolio discovery and inventory
2. Readiness assessment
3. Automated Deployments
4. Import applications
5. Transform applications
6. Validate, promote, decommission
1. PORTFOLIO DISCOVERY & INVENTORY

WHAT?

Identify and list existing assets (applications, infrastructure) in a central inventory.

HOW?

Manual

Use your favourite spreadsheet editor!

Automated

Use some tools to automate the discovery process:
- Agent(less) / Port & packet scan / SaaS or on-premise
- Self written (e.g. with Ansible) vs. commercial offerings (e.g. CloudScape)
- Use existing tools (CMDB, SOE, infrastructure as a code, CI/CD, app repository)
1. PORTFOLIO DISCOVERY & INVENTORY

Create an exhaustive list (spreadsheet) of all applications (plan, governance, estimates)

**EXAMPLE**

<table>
<thead>
<tr>
<th>Business domain</th>
<th>Logical application</th>
<th>Artifact name</th>
<th>Relevant version(s)</th>
<th>Application server</th>
<th>Type</th>
<th>Owner</th>
<th>In-scope?</th>
<th>Target release</th>
<th>Complexity</th>
<th>Business criticality</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>CoreBanking</td>
<td>bank.ear</td>
<td>5.4_2017.FINAL</td>
<td>WLS 3.4.2</td>
<td>Front-end</td>
<td>Bob Meister</td>
<td>yes</td>
<td>Q2 2021</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Core</td>
<td>CoreBanking</td>
<td>bank-backend.jar</td>
<td>5.4_2017.FINAL</td>
<td>WLS 3.4.2</td>
<td>Backend</td>
<td>Bob Meister</td>
<td>yes</td>
<td>Q2 2021</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Batch</td>
<td>JobManager</td>
<td>process-engine_war</td>
<td>12.0.3</td>
<td>WLS 3.4.6</td>
<td>3rd party</td>
<td>Jack Fruit</td>
<td>no</td>
<td>-</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Owners could be further split between ...
- technical lead / business owner / test owner / external provider
- Custom group for similar applications
- Internal application classification........
- Planning and scope
- Relevant for prioritization
CONTAINERIZATION JOURNEY

1. Portfolio discovery and inventory
2. Readiness assessment
3. Automated Deployments
4. Import applications
5. Transform applications
6. Validate, promote, decommission
2. READINESS ASSESSMENT

WHAT?

1. **Group** similar applications based on their business & technical characteristics

2. **Assess** container-readiness

3. **Identify** the right transformation based on business value

4. **Estimate** efforts

HOW?

- Conduct interviews, workshops, code & document reviews (all levels).
- **Questionnaire-based frameworks**
- **Automated code analysis tool** (e.g. RHAMT for Java applications)
## 2. READINESS ASSESSMENT

### GROUPING APPLICATIONS

#### Java EE technologies

- **Web**
  - JSF
  - JSP
  - Servlet
  - web.xml
  - WebSocket
- **Messaging**
  - JMS queue
  - JMS topic
  - JMS connection factory
- **HTTP**
  - JAX-RS
  - JAX-WS
- **Database**
  - JDBC datasource
  - JDBCXA datasource
- **Clustering**
  - JPA / Web session / EJB
- **Security**
  - Security realm
- **Transactions**
  - JTA
- **Processing**
  - Batch
  - CDI
  - JSON-P

#### Embedded frameworks

- **MVC**
  - Spring-MVC
  - Struts
  - Wicket
  - GWT
- **REST**
  - Jersey
  - Unirest
- **Web Service**
  - Axis
  - CXF
  - XFire
- **Object Mapping**
  - Hibernate
  - Hibernate OGM
  - Eclipse Link
- **Caching**
  - ehcache
  - infinispan
- **Logging**
  - log4j
- **Security**
  - ESAPI
- **Test**
  - JUnit / ...
- **Rules & Processes**
  - Drools / JBPM / iLog
- **Integration**
  - Camel / Teiid
- **3rd party**
  - Liferay / Oracle Forms /...

#### Rich applet

- JNLP

#### Markup

- HTML
  - CSS
  - JS

#### Bean

- Stateless (SLSB)
- Stateful (SFSB)
- Message (MDB)
- Managed Bean

#### Other

- JCA
- JNI
- Mail
- RMI

#### Persistence

- Persistence units
- JPA entities
- JPA named queries

#### Messaging

- JMS queue
- JMS topic
- JMS connection factory

#### Messaging

- JCA
- JNI
- Mail
- RMI

#### Other

- JCA
- JNI
- Mail
- RMI

#### Persistence

- Persistence units
- JPA entities
- JPA named queries

#### Caching

- ehcache
- infinispan

#### Object Mapping

- Hibernate
- Hibernate OGM
- Eclipse Link

#### Logging

- log4j

#### Security

- ESAPI

#### Test

- JUnit / ...

#### Rules & Processes

- Drools / JBPM / iLog

#### Integration

- Camel / Teiid

#### 3rd party

- Liferay / Oracle Forms /...
2. READINESS ASSESSMENT

CONTAINER-READINESS FACTORS

Software architecture
- Monolith vs. Modular
- Coupling (loose vs. tight)
- State management
- Communication protocols
- CPU, IO, storage, connectivity needs
- Compliance, security, availability, resiliency requirements

Dependencies
- OS & hardware
- Licensing and vendor support (3rd party components)

Processes
- Monitoring, alerting, log & configuration management

SCOPING A TRANSFORMATION

- Business criticality, usage, user base
- Application lifecycle (age)
- Expected business value & efforts
2. READINESS ASSESSMENT

TOOLING SUPPORT

RED HAT APPLICATION MIGRATION TOOLKIT

Catalyze large scale application modernizations and migrations

- Automate code analysis
- Support effort estimation
- Accelerate code migration
- Free & Open Source

Red Hat Application Migration Toolkit

- IBM WebSphere AS
- Oracle WebLogic Server
- Java EE upgrades
- JBoss EAP upgrades
- Cloud readiness, containerization
- Pluggable: your own rules

- Command line interface
- Web console
- Eclipse plugin

#redhat #rhsummit
2. READINESS ASSESSMENT

TOOLING SUPPORT

Download the Toolkit on https://developers.redhat.com/products/rhamt/overview/
DEMO
Technologies

eureka-ear-2.0.ear

View
Connect
Store
Sustain
Execute

Java EE
- Web
  - JSP Page: 20
  - Web XML File: 3
- Other
  - Mail: 25
- Persistence
  - Persistence units: 2
  - JPA entities: 51
- Security
  - Security Realm: 22
- Processing
  - CDI: 3

Embedded
- Markup
  - CSS: 300
  - JavaScript: 444
  - HTML: 54
- REST
  - jersey: 9
- Object Mapping
  - Hibernate: 6
- Caching
  - ehcache: 2
- Logging
  - SLF4j: 3
  - Apache Log4j: 2
  - Apache Commons Logging: 2
- Rules & Processes
  - Drools: 2
- Inversion of Control
  - AOP Alliance: 2
2. READINESS ASSESSMENT

RESULTS & PRIORITIZATION

Container Readiness

Business value

Containerization status
- Done!
- In progress
- Not started
- Out-of-scope!

Level-of-Efforts
CONTAINERIZATION JOURNEY

1. Portfolio discovery and inventory
2. Readiness assessment
3. Automated Deployments
4. Import applications
5. Transform applications
6. Validate, promote, decommission
3. AUTOMATED DEPLOYMENTS

WHAT?
● Design the new platform (MVP, requirements)
● Document architecture decisions.
● Build the full-ecosystem

HOW?
Adapt push-button infrastructure (reference architecture)
● All needed stack to start building solutions on day one
● Robust IaaS, PaaS, DevOps toolchain & AppDev services
● Fully-automated deployment

https://www.redhat.com/en/explore/my-open-innovation-lab-stack
CONTAINERIZATION JOURNEY

1. Portfolio discovery and inventory
2. Readiness assessment
3. Automated Deployments
4. Import applications
5. Transform application code
6. Validate, promote, decommission
4. IMPORT APPLICATIONS

WHAT?

Bridge existing and new platforms
- Import assets of the existing application lifecycle (source code, configuration, repositories, processes)
- Reduce manual migration gaps

HOW?

- Automate import of all applications (no code change)
  - Provide a “deploy-to-new-platform” button to devs
  - Pre-generate templates and configuration
- Enable users (templates, guides)
CONTAINERIZATION JOURNEY

1. Portfolio discovery and inventory
2. Readiness assessment
3. Automated Deployments
4. Import applications
5. Transform applications
6. Validate, promote, decommission
5. TRANSFORM APPLICATIONS

WHAT?

Update code & configuration to containerize an application according the chosen transformation.

HOW?

Step-by-step approach with well-defined scope & activities. As few changes as necessary.

1. Image choice, initial configuration
2. Pre-emptive code changes (e.g. RHAMT)
3. Get application running in a container
4. Optional changes (refactor, strangle)
5. Test & Promote
CONTAINERIZATION JOURNEY

1. Portfolio discovery and inventory
2. Readiness assessment
3. Automated Deployments
4. Import applications
5. Transform applications
6. Validate, promote, decommission
6. VALIDATE, PROMOTE, DECOMMISSION

WHAT?

- Development Workflow from git push to deployment
- Approvals, Checks, Sign-offs, Audit trails etc.
- Predictable, measurable and repeatable methods

HOW?

- Automated testing
- CI/CD pipeline (S2I)
- Automated Software Supply Chain
- A platform that gives you most of it OOTB
DEPLOYMENT PIPELINES

1. DEVELOPER ➔ GIT SERVER ➔ ARTIFACT REPOSITORY
2. OPENSHIFT CI/CD PIPELINE (JENKINS)
   - IMAGE BUILD & DEPLOY
   - PROMOTE TO TEST
   - PROMOTE TO UAT
   - PROMOTE TO PROD
3. INTEGRATED IMAGE REGISTRY
4. OPENSHIFT CLUSTER
   - NON-PROD
     - DEV
     - TEST
     - UAT
   - PROD
5. RELEASE MANAGER
   - GO LIVE?
THE REAL CHALLENGE
CONTAINERIZATION JOURNEY

1. Portfolio discovery and inventory
2. Readiness assessment
3. Automated Deployments
4. Import applications
5. Transform applications
6. Validate, promote, decommission
CONTAINERIZATION JOURNEY AT SCALE!

**DISCOVER**
- Portfolio discovery and inventory

**DESIGN**
- Solve & document typical issues
- Pilot applications
- Automated deployments
- Import applications

**SCALE**
- Transform applications
- Validate, promote, decommission

FACTORY with ITERATIONS
NEXT STEPS
FURTHER READINGS

https://developers.redhat.com/resources/#!type=book
HEAD TO DISCOVERY ZONE!

How to build a successful onboarding program for OpenShift
Eric Sauer, Red Hat, Inc.; Malcolm Herbert, Red Hat

Description
Many organizations succeed in getting OpenShift deployed and operationalized, but then struggle getting their application teams to adopt. In this interactive session, Red Hat consultants will assess the landscape of the audience’s platforms, and using this data, present strategies to build an onboarding program. Focusing first on strong early adopter candidates, then scaling out by turning early success into repeatable processes, you’ll walk away with best practices to bring to the rest of your organization.

Time
Tuesday, May 8 3:30PM - 4:15PM

Location
Customer Success Zone, Moscone West, Level 2

Speakers
Eric Sauer
Red Hat, Inc.

Decompose a monolith with microservices
David Gordon, Red Hat; Michael Costello, Red Hat

Description
As organizations look to decompose their monoliths and introduce a microservices architecture, microservices can begin to form dependencies on each other. Just one service can cause a cascade of problems, like a domino effect, causing total service outages in the worst case scenario. In this interactive session, Red Hat consultants go beyond container orchestrations, autoscaling, and traditional Platform-as-a-Service features and dive into best practices to ensure integration platforms at scale don’t come to a screeching halt. In this discussion, we’ll highlight and demonstrate service mesh, and circuit breaker patterns to ensure stability and consistency in our APIs and microservices architecture using OpenShift.

Time
Thursday, May 10 3:00PM - 3:45PM

Location
Customer Success Zone, Moscone West, Level 2

Speakers
David Gordon
Red Hat

... and lots more ....
THANK YOU

Zohaib Khan
App Modernization Lead
Red Hat - NAPS
zkhan@redhat.com
@zeebluejay

Marc Zottner
AMM Program Manager
Red Hat - EMEA
mzottner@redhat.com

plus.google.com/+RedHat
linkedin.com/company/red-hat
youtube.com/user/RedHatVideos
facebook.com/redhatinc
twitter.com/RedHat
CONTAINERIZATION IN PRACTICE

Parallel Dev/Ops workstreams
- Devs don’t need to wait for the final infrastructure (OCP CDK)

Plan
- Containerize pilot applications
- Automate import (apps & conf)
- Secure & accelerate population of the new platform

Application development
- Proof of concepts
- Design & build new infrastructure

Infrastructure operations
- Assessment
- Parallel Dev/Ops workstreams
WHY IS OPENSHIFT THE BEST PLATFORM?
THE FOUR Cs

**CODE**
Kubernetes commits

- Red Hat: 46%
- Google: 18%
- Independent: 19%
- Others: 17%

**CUSTOMERS**
Most reference customers running in production.
We have years of experience running OpenShift Online & OpenShift Dedicated services.

**CLOUD**
Strong partnerships with cloud providers, ISVs, CCSPs.
We have an extensive container catalog of certified partner images.

**COMPREHENSIVE**
Our comprehensive portfolio of container products and services includes developer tools, security, application services, storage, & management.

---

Red Hat is a leading Kubernetes developer & contributor with Google\(^1\).

We make container development easy, reliable, & more secure.

---