Welcome!

2019 NASWA Workforce Summit
& UI Directors’ Conference with IT/Legal Issues Forum

83rd Annual Meeting

Revolutionizing the Workforce System
Boston, Massachusetts
Microservices and Containers and Use in WF Domain

Lou Ansaldi
NASWA Technical Director
Who Uses

- Netflix
- eBay
- Amazon
- Twitter
- PayPal
- Many More
What are microservices

• Departure from monolithic application
  • Built as a single, autonomous, BULKY unit
  • App changes typically affect entire code base
    • Likely need to build and deploy entire app
      • Sluggish, not nimble
  • Unintended dependencies create vulnerabilities
    • Fixes can break code elsewhere
  • Testing expansive and laborious

• Collection of small services
  • Run in its own process
  • May use different data storage techniques
  • Connect via APIs
    • OpenAPI (Swagger) file defines the API, post and gets and their input, output parms
    • Leverage RESTful and web service ecosystem techniques
      • Tools and solutions
    • Testing these APIs can help validate the flow of data and information throughout your microservice deployment.

Promotes real object oriented separation of concerns
Myriad of Advantages

• Code can be independently developed
  • Use whatever technology best fit for delivering given function
  • **Technology heterogeneity** or polyglot programming
    • Different part of systems can use different technology stacks if needed for better performance.
    • However, overhead of understanding of multiple technology stacks.
    • Address through governance and standards across teams.
  • Compliments agile teams and squads

<table>
<thead>
<tr>
<th>Simpler To Deploy</th>
<th>Deploy in literal parts without affecting other services.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simpler To Understand</td>
<td>Follow code easier since the function is isolated and less dependent.</td>
</tr>
<tr>
<td>Reusability Across Business</td>
<td>Share small services like payment or login systems across the business.</td>
</tr>
<tr>
<td>Faster Defect Isolation</td>
<td>When a test fails or service goes down, isolate it quickly with microservices.</td>
</tr>
<tr>
<td>Minimized Risk Of Change</td>
<td>Avoid locking in technologies or languages - change on the fly without risk.</td>
</tr>
</tbody>
</table>
Myriad of Advantages - concluded

• Resiliency
  • If one microservice inoperable system may still operate

• Independent scaling

• Composability
  • Seamless replacement of a service

Makes plug n play no longer a Myth or Promise
Beware Though

• If Not Carefully Conceived, Complexity may escalate
• Many tools still not designed to work with service dependencies
• Increases tech stacks can cause the application difficulty in maintenance
• As each service has its own database, transaction management and data consistency may become very challenging
• Each service has to be testing and monitored increasing demand for automation
• The initial refactoring of a monolithic application can be exceedingly complex for large enterprise applications
• Security Challenges
Microservice Patterns

• Decomposition by business function
• Strangler
• API Gateway
• Aggregation of Data
• Saga
  • Ensures data consistency
  • Idempotency
Containers

- Containers are a logical packaging of Applications abstracted from the run environment
- Containers virtualize at the operating system level
  - Instead of virtualizing the hardware stack as with the virtual machines approach
  - Multiple containers running atop the OS kernel directly.
  - Therefore containers are far more lightweight
    - Share the OS kernel
    - Start much faster
    - Use a fraction of the memory compared to booting an entire OS
## Containers vs. Virtualization

<table>
<thead>
<tr>
<th></th>
<th>CONTAINER BENEFITS</th>
<th>VIRTUAL MACHINE BENEFITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consistent Runtime Environment</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Application Sandboxing</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Small Size on Disk</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>Low Overhead</td>
<td>YES</td>
<td></td>
</tr>
</tbody>
</table>
Container Advantages

• **Google: Gmail to YouTube to Search**
  • Portability between different platforms and clouds—it’s truly write once, run anywhere.
    • Even if target environment is a private data center, the public cloud, or even a developer’s personal laptop.
  • Increased Efficiency
    • Use far fewer resources than VMs
    • Higher utilization of compute resources.
• Developer Agility as can integrate with their existing DevOps environment.
• Higher speed in the delivery of enhancements
• Improved security by isolating applications from the host system and from each other.
• Faster app start-up and easier scaling
• Flexibility to on virtualized infrastructures or on bare metal servers
• Easier management since install, upgrade, and rollback processes are built into the Kubernetes platform.
Not All Container Technologies are Equal
OpenShift and OKD vs Kubernetes

• Kubernetes Integral to OpenShift
• OpenShift Platform is a Product vs. Kubernetes is Project
  • For Platform pay more when cluster grows
• Kubernetes ahead of OpenShift
• Support Differs
• Kubernetes installs on more Linus Distributions (e.g., Debian, Ubuntu)
• OpenShift more strict security policies than Kubernetes out-of-the-box
• Kubernetes Helm Charts much more flexible than OpenShift Templates
Microservices and Containers in the WF Domain

• Critical Areas
  • Right Sizing of Monolithic Base Systems
  • Data ownership
    • Fed Reporting
  • Security
    • Including Role-Based implementations
• Still need
  • Robust testing
    • Unit, component, etc
    • Security testing
    • Automated Regression
    • Performance
  • Code Quality
  • Etc
Microservices and Containers

- iUS

- ReEmploy
  - ES and BIRT
  - Challenge business layer decomposition
    - Mon ReDet
  - ICON Relay
  - Three (3) to Five (5) Migration
    - Part of InnovateUI

- Many Other States Interested
BACKUP

• Consistent Environment
• Generally Run Anywhere
• Isolation