Are Containers the New Golden Hammer?
Tracy Bannon
Lori Olson
Agenda

1. Introduction

2. Solution Anti-Patterns and How to Avoid Them
   A. Anti-Pattern #1 – Containers as an Alternate Virtual Machine (VM)
   B. Anti-Pattern #2 – Pushing the Architecture before the Organization is Ready
   C. Anti-Pattern #3 – Lack of Layering Strategy

3. Conclusion – The Golden Hammer Anti-Pattern
Deloitte Corporate Profile Overview

"Deloitte" is the brand under which dedicated professionals in independent member firms throughout the world collaborate to provide audit, consulting, financial advisory, risk management and tax services to clients.*

For more than 150 years, clients have relied on Deloitte to serve their ever-changing needs. A provider of financial and economic advisory services around the world, Deloitte has conducted over 1,300 technical assistance engagements in over 100 countries. We are a national and global leader today because, throughout our history, we have continuously sustained our clients’ trust and exceeded their expectations. Our renowned depth of technological and cross-industry knowledge, combined with our ability to seamlessly and strategically integrate solutions, has allowed us to aid our client’s ability to achieve meaningful impact towards their overall mission.

Our independent country firms are members of Deloitte Touche Tohmatsu Limited (DTTL), a UK private company limited by guarantee. Each member firm provides services in a particular geographic area and is subject to the laws and professional regulations of the particular country or countries in which it operates. Each DTTL member firm is structured differently in accordance with national laws, regulations, customary practice, and other factors, and may secure the provision of professional services in its territory through subsidiaries, affiliates, and/or other entities.

*Certain services may not be available to attest clients under the rules and regulations of public accounting. Please see www.deloitte.com/about to learn more about our global network of member firms.
Poll

Web Browser @ [PollEv.com/rockitect](http://PollEv.com/rockitect)
or

Text 39712 and your message to 22333
How are you leveraging containers?

reusable jumpstarts
Background

The container ecosystem is growing rapidly and there is a rush to adopt this technology. Many enterprise implementations run the risk of this being a golden hammer by applying it whenever and wherever possible. Join us as we dive deep to establish a common taxonomy and definition of containers, investigate practical use cases, and identify key components and decisions points for creating your container strategy.

Participants in this session will learn:

- What containers solve and what they do not solve
- Key concepts for architecting containers and the decision for layering containers
- Organizational recommendations for effective adoption
- How to manage and maintain your containers – tools can help but don't take the place of your strategy. Even with tools, it's not as simple as "just change a layer."
- Practical tips from the field
Anti-Pattern #1:
Containers as an Alternate Virtual Machine (VM)
Anti-Pattern #1: Containers as an Alternate Virtual Machine (VM)

VMs and Containers are unique, not interchangeable

*Treat VMs and Containers as separate concepts*

- Decreasing operating efficiency
- Security vulnerabilities
- Complex process and configuration management

<table>
<thead>
<tr>
<th>Use VMs to:</th>
<th>Use Containers to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Virtualize operating systems not supported by containers</td>
<td>• Support infrastructure as code</td>
</tr>
<tr>
<td>• Provide general purpose operating system features</td>
<td>• Speed deployment and lightweight execution</td>
</tr>
</tbody>
</table>

Virtual Machines

- Hypervisor
- Server
- Guest OS

Containers

- OS Kernel
- Server
- Containers

Use VMs to: Use Containers to:

- Support infrastructure as code
- Speed deployment and lightweight execution
- Deploy across hybrid environments and avoid cloud vendor lock-in
Leading Container Practices
Getting the most out of your compute resources

**Define lean images**
- Start small
- Write smart logic to prevent unnecessary caching of resources

**Eliminate what you don’t use**
- Use the host operating system to run apps
- Avoid running unnecessary services
- Install only necessary libraries

**Construct container layers with purpose and understanding**
- Build stateless light-weight containers
- Deliberately define layers
- Reuse common layers where possible
- Understand consequences of layer selection
Anti-Pattern #2: Pushing Architecture before the Organization is Ready
Ant-Pattern #2: Pushing the Architecture before the Organization is Ready

**Signs your organization may not be ready**

- Organizational structure doesn’t align teams to features
- Teams aren’t empowered
- Lack of Automation/DevSecOps processes

**Signs your people are not ready**

- Overly complex containers
- Bloated containers
- Simple facades with complex deployment patterns
- General sloppy container design
Empowering Adoption
Steps to prepping your organization and people

- Organization is structured around BFF (business feature focus)
- Automation mindset and abilities are table stakes
- Teams are educated on container design patterns
- Just Enough Governance and a focus on centralized guidance, decentralized execution

Layer Focused Teams
- UI Specialists
- Middleware Specialists
- DBAs

Feature Focused Teams
- Shopping Cart Team
- Catalog Team
- My Account Team
Anti-Pattern #3:
Lack of Layering Strategy
Anti-Pattern #3: Lack of Layering Strategy

**Lack of a layering strategy may introduce:**

- Instability from poor version management
- Deployment complexity from multiple dependencies
- Performance impacts from fine-grained containers
- Security exposure from uncontrolled layers

**A well-defined layering strategy should:**

- Explicitly define dependencies
- Decrease proliferation, which can result in less complexity
- Allow for optimal maintainability with the right level of granularity
- Secure and standardize layers with managed repositories
### Container Layering Strategy and Guidelines

Decisions to make in order to have a good layering strategy

<table>
<thead>
<tr>
<th>Inventory required assets</th>
<th>Set up guidelines</th>
<th>Explicitly define dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Determine base image</td>
<td>- Do not change underlying base/composition level image</td>
<td></td>
</tr>
<tr>
<td>- Determine composition layer images (CentOS + Oracle JDK)</td>
<td>- e.g. If Java-based, use image ABC; if Python-based, use image XYZ</td>
<td></td>
</tr>
<tr>
<td>- Determine final containers (CentOS+OracleJDK) + SonarQube)</td>
<td></td>
<td>- Decreases proliferation, which can result in less complexity (decreasing number of distinct containers that are used)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Allows for optimal maintainability with the right level of granularity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Secures and standardizes layers with managed repositories</td>
</tr>
</tbody>
</table>

**Tools that help you do this:** Ansible and OpenShift
Conclusion:
The Golden Hammer Anti-Pattern
Conclusion
Golden Hammer Anti-Pattern

Containers are not a ubiquitous solution

Misuse may lead to:
- Consumption of unneeded resources (compute, storage, network overhead)
- Significant costs debugging and maintaining code
- Unnecessary complexity

Containers can provide the most benefit when teams or systems have:
- New development when you can architect for container
- High-elasticity needs
- Business demand for rapidly changing feature sets
- Feature-team alignment
- Loosely coupled sets of APIs
- Well-defined seams where there are not remote dependencies
Thank You