Microservices – Is it the Holy Grail?

A Perspective from a Developer
Bettina Bassermann  
Product Sales Specialist Cloud & DevOps  
Bettina.Bassermann@suse.com

Rania Mohamed  
Solution Architect, Services Consulting  
Rania.Mohamed@suse.com
Who is SUSE?

- Founded in 1992
- Largest independent open source vendor as of March 2019
- Technology company
- Our Mission is to help customers to master the digital transformation through Open Source technology
- Innovating with Partners and communities
- Enterprise-Grade Support
Series about modern Application Development

• Software Development, Microservices & Container Management, a SUSE webinar series on modern Application Development
• Please find all SUSE Webinars here
  
  https://www.suse.com/de-de/events/webinars

  Microservices – Is it the Holy Grain? A Perspective of a Developer
  Container and Cloud Native Technologies – Why do we need them and what is so great about it?
  Why Kubernetes? A Deep Dive in Options, Benefits and Usecases
  About making Choices – CaaS Pv4 as SUSE’s empowering of Kubernetes

  ....stay tuned for the 2020 sessions with the Chamelion
Agenda

• Evolution of Software and App Development
• What is Microservices and what not?
• Microservices vs Container
• Microservices vs SOA
• Microservices vs Cloud Native Applications
• Benefits and Challenges – in Business and Technology
• How to overcome Microservices challenges?
• When to go Microservices and when not to go for Microservices?
Evolution of Software and Application Development

Software Development started by big monolithic applications, growing into beasts.
What is Monolith CLic App?
Evolution of Software and Application Development - Monolithic App

- A **big fat** application
- **Self contained.**
- **No separation of concerns** (business context and technical context)
- **All one unit**
- **Structure programing.**
- **Interconnected and interdependent** (tightly coupled)
- Originated by **mainframe** apps.
- **Hard** to maintain, manage and test.
Software Development
Modern Techniques
Evolution of Software and Application Development - Modern Techniques

- Application modularity
- Object oriented – OO
- Better governance
- Better maintainability
- A bit of Separation of Concerns

But it is still mono!
Evolution of Software and Application Development - Modern Techniques

- **Multi-tier** application architecture
- **Mono** is broken into small components based Architecture (modules and tiers)
- **Service oriented architecture** (SOA)
- **Flexible** design principles
- **De-coupling** the services
- **Reusability** (no code redundancy)
- **Well defined contracts** (APIs centric)
- **Service discoverability**
- **Autonomous**

But **complex, expensive, hard to scale & maintain**
Component based application example

Client Device → Load Balancer → User Interface

Client Device → Load Balancer → Business Logic

Database Cluster
Evolution of Software and Application Development

- High agility
- High scalability
- High adaptability
- Digitalization Business

Micro & Nano Development Approaches
What is Micro & Nano Software Development?
What is Microservices?
Evolution of Software and Application Development – Microservices

- Smallest deployable unit that can go by itself
- Self-contained
- Finest grained unit (business and technical)
- Highly loosely coupled
- Independent from the external world.
- Serves a fine business needs
- Doesn’t deliver business
- Deliver business by orchestration.
- Enables/implements decentralization
- Single source of truth for a fine business feature.

It doesn’t care if the whole external world is in war and down 😊
MSA example

Client Device

Load Balancer

Application Hosting Runtime

MSA1

Micro UI

Micro business feature

MSA2

MSAn
What is Microfunctions?
Evolution of Software and Application Development - Microfunction

- Serverless development
- No dedicated server/platform to run it
- Not always running but always available
- Instance gets available as per the need.
- Cost is by actual calls/invocation to the function 😊.
- Much more smaller in its footprint
- A very short lifetime from an execution and processing perspective.
- No database
- May communicate with database and datastores.
- Stateless
- Event based architecture (Complex Event Processing)
- Helps in digitalizing business.

Scales to zero instances if no requests are sent to the function 😊, it is the magic of serverless
What is Cloud Native Development?
Evolution of Software and Application Development – Cloud Native Development

- Is a **Microservice** (but not the other way around 😊)
- **Portable & Agile**
- **Natively consuming** the underlying cloud platform
- Runs on **PaaS offering runtime**
- **Stateless**
- **Polyglot**
- **Event driven architecture**
- **Enable smooth integration**
- **High SLA**

“I want to write my code, and run it in the cloud, I don’t care how 😊.”
Evolution of Software and Application Development - Summary

**Big Monolithic App (mainframe)**
- Hard to maintain/manage/support
- Very expensive

**Multi-tier App (Separation of Concern technical wise)**
- Better from Maintenance
- Still Mono, expensive to scale

**Service App (SOA) centric**
- Not Mono Anymore
- Complex in management
- Expensive
- Loosely couple

**Micro-Functions (Serverless)**

**Modular Monolithic App (Separation of Concern business wise)**
- Better from Maintenance
- Still Mono, expensive to scale

**Component based (Modules and tiers) App (Separation of Concerns technical and business wise)**
- Broken the mono into components a but not granular
- Still tightly coupled
- Expensive to scale and test

**Micro-services (MSA)**
- Ease of efficient scalability
- High agility

**Cloud-Native Application (CNA)**
What is MSA and What is not MSA?
What is Microservices and what not? – MSA main Design/Architecture Principles

• Owns its data storage
• Independent on others (data & APIs).
• May be stateful or stateless
• State must be properly persistent and managed.
• Fine grained (actually the finest 😊)
What is Microservices and what not? – MSA Example
When to go Microservices and when not to go for Microservices? – Misunderstandings for MSA

• 12 factor app, is not MSA
• Not an SOA alternative
• Not an SOA app
• Not just breaking up an app into a set of services.
• Not just API-centric architecture
• Not DevOps
• Virtualizing (VM) the app Running code is not MSA
• Containerizing the app is not an MSA
• Using API gateway is not MSA
What is Microservices and what not? – Non MSA Example
Microservices versus Container
## Microservices Vs Containers

<table>
<thead>
<tr>
<th>Containers</th>
<th>MSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Lightweight standalone app + its dependencies</td>
<td>- MSA can be developed as a set of containers or not</td>
</tr>
<tr>
<td>- Smaller footprint than a VM</td>
<td>- Usually hosts the database and storage as a backing service</td>
</tr>
<tr>
<td>- Very minimal Operating system</td>
<td>- Uses Events for Integration</td>
</tr>
<tr>
<td>- May embed the app database and storage</td>
<td>- Uses Service brokers for backing service provisioning</td>
</tr>
<tr>
<td>- Required orchestration engine such as K8s.</td>
<td></td>
</tr>
<tr>
<td>- Integration more using the services offered by orchestration engine.</td>
<td></td>
</tr>
</tbody>
</table>
Microservices Vs Containers

Containers

Host Operating System
BareMetal Server
VMs
Containers

Pod1
App1
App2
App3
App4
App1
App2
App3

Deployment

Event Bus
PaaS Runtime or Container Orchestrator
Service Broker
Host Operating System
Cloud Or BareMetal

MSA

Container as a Service

Container Engine
Host Operating System
BareMetal Server

MSA1
MSA2
MSA3
Microservices versus SOA
# Microservices Vs SOA

<table>
<thead>
<tr>
<th>SOA</th>
<th>MSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Complex</td>
<td>• Simple</td>
</tr>
<tr>
<td>• More driven by services</td>
<td>• More driven by business capabilities</td>
</tr>
<tr>
<td>• More into coarse grained services</td>
<td>• More fine grained apps</td>
</tr>
<tr>
<td>• API centric architecture</td>
<td>• More into event driven architecture</td>
</tr>
<tr>
<td>• Expensive</td>
<td>• Cost efficient</td>
</tr>
<tr>
<td>• Hard to scale horizontally</td>
<td>• Ease to scale horizontally</td>
</tr>
<tr>
<td>• Solves integration and orchestration/ choreography problems</td>
<td>• Solves business agility and scalability</td>
</tr>
<tr>
<td>• Centralized architecture</td>
<td>• Decentralized architecture</td>
</tr>
<tr>
<td>• Hard to implement resiliency</td>
<td>• Resiliency is indented by design for failure principle</td>
</tr>
</tbody>
</table>

MSA is not replacing SOA, they solve different problems
Microservices versus Cloud Native Applications
## Microservices Vs Cloud Native Applications

<table>
<thead>
<tr>
<th>CNA</th>
<th>MSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Must be in the <strong>cloud</strong></td>
<td>• May be hosted in the <strong>cloud</strong> or not</td>
</tr>
<tr>
<td>• Is an <strong>MSA</strong></td>
<td>• May or may not be a <strong>CNA</strong></td>
</tr>
<tr>
<td>• Doesn’t care for the app runtime</td>
<td>• May host its own <strong>runtime</strong></td>
</tr>
<tr>
<td>• No embedded storage → backing service (dependency)</td>
<td>• May hosts its own <strong>data storage</strong></td>
</tr>
<tr>
<td>• Must be <strong>stateless</strong></td>
<td>• May be <strong>stateful or stateless</strong></td>
</tr>
<tr>
<td>• Must uses to underlying <strong>Cloud natively</strong> for handling an event (e.g., failure of a function)</td>
<td></td>
</tr>
<tr>
<td>• May be used to build <strong>SaaS</strong></td>
<td></td>
</tr>
</tbody>
</table>
Microservices Vs Cloud Native Applications

Container as a Service

Cloud Native Application

- MSA1
- MSA2
- MSA3

Event Bus

PaaS Runtime or Container Orchestrator

Service Broker

Host Operating System

Cloud Or BareMetal
Benefits and Challenges – in Business and Technology
MSA Benefits

- Ease of Development & Maintenance
- Ease of Scale out/in
- Better handling to variable load
- Cost Efficient resources
- Designed and distributed around business capabilities
- Agility
- Consistency
- Autonomous
- High Resiliency/ Fault isolation
- Enables self healing

- Enables business and technical innovation
- Enables Real time apps
- Supports time to market
- Enables rapid Business services Growth
- Enhance quality
- Enables high SLAs
- Improves development productivity
MSA Challenges

• Design MSA
• MSA Integration and Collaboration
• Integration Testing
• Managing & Monitoring (an app can have 400+ MSA)
• Implicit Cyclic dependencies
• Fault Tolerance
• Data Synchronizations
• Security
• Discovery
• Logging
• Business Troubleshooting
• Governance
How to overcome Microservices challenges?
How to overcome Microservices challenges?

- Always design for failure
- 100% Automation
- Integrate DevOps (CI/CD/CD)
- Use Central logging and Monitoring
- Implement service mesh for governance
- Implement log analytics and system analytics → high responsive and proactive system
- Implement MSA known design patterns when needed (e.g., service discovery, data versioning, eventual consistency…)
- Implement integration patterns (e.g. aggregator, circuit breaker) using orchestrator and event processing systems
- Use Function as a service for integration and in conjunction with MSA to deliver the E2E solution.
- Use API Gateway for securing MSAs
When to go Microservices and when not to go for Microservices?
When to go MSA and when not to go for MSA?

Go To MSA

• Variable Load
• Frequent changes are required
• Time to market is an objective
• It is already complex
• Building a PaaS or a SaaS
• Targeting multi-cloud
• Targeting Digital Transformation

Don’t Go to MSA

• Load is predictable
• No changes in the business (it is mostly stable)
• No integration with the external world
• Simple or moderate application

Don’t build a Rocket to drive you home from the office 😊
SUSE: Underpinning Digital Transformation

**Infrastructure & Lifecycle Management**
- SUSE Manager

**Software-defined Infrastructure**
- Private Cloud / IaaS
  - SUSE OpenStack Cloud
- Compute
  - Virtual Machine & Container
- Storage
  - SUSE Enterprise Storage
- Networking
  - SDN and NFV
- Multimodal Operating System
  - SUSE Linux Enterprise Server
- Physical Infrastructure: Multi-platform Servers, Switches, Storage

**Application Delivery**
- Container Management
  - SUSE CaaS Platform
- Platform as a Service
  - SUSE Cloud Application Platform

**Open, Secure, Proven**

**Services**
- SUSE Global Services
- Consulting Services
- Select Services
- Premium Support Services
Q&A
Thank you
Unpublished Work of SUSE LLC. All Rights Reserved.
This work is an unpublished work and contains confidential, proprietary and trade secret information of SUSE LLC. Access to this work is restricted to SUSE employees who have a need to know to perform tasks within the scope of their assignments. No part of this work may be practiced, performed, copied, distributed, revised, modified, translated, abridged, condensed, expanded, collected, or adapted without the prior written consent of SUSE. Any use or exploitation of this work without authorization could subject the perpetrator to criminal and civil liability.

General Disclaimer
This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. SUSE makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. The development, release, and timing of features or functionality described for SUSE products remains at the sole discretion of SUSE. Further, SUSE reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All SUSE marks referenced in this presentation are trademarks or registered trademarks of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.