AGENDA

1. *Introducing Microservices*

2. Microservices Best Practices

3. Microservices support in the Digital Business Platform

4. Roadmap
THE NEED FOR SPEED

Cloud, Mobile, Social, BigData, Blockchain, IoT, Bimodal, DevOps, Deep Learning, Containers

- cars.com: 300 Deployments / Year
- Flickr: 10+ Deployments / Day
- Etsy: 50-60 Deployments / Day
- Amazon: Every 11.6 seconds
- Kuehne+Nagel: ~50 Deployments / Day
SOA AND MICROSERVICES GOALS

SOA

• Reuse
• Composition

Microservices

• Agility
• Scalability
New architectural style for web-scale digital applications

Microservices

- Independently Deployable
- Domain Driven Design
- And
- Scalable Services
- Responsive And Resilient
- DevOps First
- Organization Alignment
THE NEW ARCHITECTURAL PARADIGM

- Microservices
- Containers
- APIs
- Continuous Delivery
- Cloud
- DevOps
- Events
1. Introducing Microservices

2. Microservices Best Practices

3. Microservices support in the Digital Business Platform

4. Roadmap
Are You Ready for Microservices?

Yes...
✓ Increase agility
✓ Enable extensibility
✓ Facilitate distributed development
✓ Support massive scalability
✓ Build (almost) unbreakable systems

... But
- Highly complex architecture
- Requires unfamiliar design patterns
- Requires new enabling infrastructure
- Requires new DevOps processes and practices
- Requires design and organizational discipline
Determine if you really need Microservices

1. Every Application is not complicated enough to warrant being broken down into Microservices.
2. Martin Fowler\(^2\) talks about “Microservice premium” where complexity of microservices hampers development productivity

Adopt DevOps and Continuous delivery

1. Continuous delivery and automation are important for microservices.
2. Small agile teams adding new features and deploy their code more frequently is an important requirement for microservices

Start with the monolith

1. Monolith is the 1\(^{st}\) step towards designing a microservices architecture
2. Based on usage and architecture principles, this application can be broken into microservices.
MICROSERVICES BEST PRACTICES

Create a shared repository for all services

1. Number of services is higher
2. Many teams are building micro-services
3. There is a need for a Service Catalogue (Service metadata, Running instances, Service dependencies, ownership and usage)

Apply comprehensive monitoring

1. Due to the distributed nature of microservices architecture, comprehensive monitoring is needed (transaction monitoring)
2. Auditing per service
WEB SCALE ARCHITECTURE
DEVOPS AND CONTINUOUS DELIVERY

- Increased agility
- Facilitates distributed development
- Enable continuous delivery
- Support massive scalability

- Daily testing of builds
- Quick capture of errors and quick fixes
- Continuous integration

- Improved platform configuration consistency
- Faster image loading
- Container runtime, clustering and abstraction layer
- Increased workload density

- Rapid and Automated Provisioning of resources
- Elastic Scalability
- Optimized resource utilization

Develop Test Package Deploy

Microservices

Automated Testing

Cloud

docker
ARCHITECTURAL PATTERNS

Traditional Integration Architecture

- Consumers
- API
- Composition & Integration Logic
- Packages (P)
- Connectivity
- Textbook Microservices Architecture
- Consumers
- API (Mini Service Layer)
- Event Bus
- Container(s)
- Hybrid Microservices Architecture
- Consumers
- API
- Event Bus
- Integration Bus
- Database
- Connectivity
- Containers (M, S)
- Dependencies (PPPP)

Packaged Apps, Big Data, Legacy, SaaS
HYBRID MICROSERVICES ARCHITECTURE

• Mix of traditional integration and microservices architecture

• Implement microservices for common functionality like logging, auditing, analytics etc.

• Independent packages (containing microservices) share the container infrastructure but can be developed by different teams

• Microservices can access backend applications and databases directly OR via the integration (bus) layer
AGENDA

1. Introducing Microservices

2. Microservices Best Practices

3. *Microservices support in the Digital Business Platform*

4. Roadmap
GARTNER REFERENCE ARCHITECTURE FOR MICROSERVICES
MAPPED TO SOFTWARE AG COMPONENTS

Additional Components

- webMethods DevOps Edition (wM TestSuite)
- Command Central for continuous integration and continuous delivery
- Integration with Consul Service Registry
- Native support for Event Driven Architecture