CHANGEPOND

DECODING MICROSERVICES

WHITE PAPER

With digital technology driving economies, companies are forced to stay agile and keep up pace like never before. As business strategies become relevant & irrelevant quickly, there is a need to flex and scale software systems which play a key role at the heart of today's businesses. The relative inefficiency of traditional web applications and other older enterprise systems has paved way for Microservice architecture.

What are Microservices?

They are a new paradigm of designing a software application as suites of independently deployable services.

These services offer organizations a great deal of potential for agility and cost reduction due to their granularity and reuse. They are based on RESTful services that are built and operated by cross-functional teams and are intended to be completely independent of the underlying platforms and applications. With this, it eschews many of monolithic web architecture's problems that can create technical debt, and in turn, brings measurable savings by both scaling and reduced time-to-market.

Microservices tends to refer to developing functionality as a collection of small services, each running in its own process and accessed via a lightweight interface, such as an HTTP RESTful API.

-Martin Fowler

Typical pain points faced by businesses in handling with legacy monolithic application:

Undocumented large application base code	Difficulty in integrating with current modular and flexible IT architectures	Piled up technology debt and lack of platform support
Expensive to maintain	Less frequency in updates & functionalities	Inability to scale the application

Are Microservices right for me?

The Microservice design paradigm is growing in popularity and it is tempting to think that Microservices are the silver bullet for your organization's IT needs.

The end goal of any organization's IT strategy or modernisation program should be to ***deliver value to both internal stakeholders and customers.***

From an IT perspective, this is about ***delivering safer, more rapid changes to your software systems.***

Before getting into Microservices you should evaluate your value stream (i.e. any business activity that has to happen from idea creation to production).

Value stream analysis will help you identify where the bottlenecks are in your processes. These, in turn, will help you determine whether Microservices are the best fit solution for your organization.

Microservice Ecosystem market will grow globally at a robust CAGR 16% between 2017 and 2022, reaching $10 billion by 2021

– www.marketanalysis.com

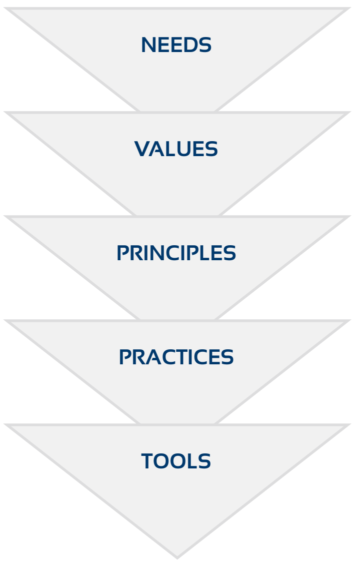
What are the most important things to consider before building Microservices?

Business Goals and Strategy

Your Microservices should be driven by business functionality and goals.

Ensure your business goals are SMART (Specific, Measurable, Achievable, Relevant and Time-bound) and the accompanying strategies are well defined. It is key to communicate the vision with all stakeholders.

Continuously validate and map these goals against architectural principles and design/delivery practices.



Strategic Goals

Support entry into new(global) markets

- Global platform, development and configuration
- English-speaking markets are primary target

Support innovation in existing markets

- Reduce cost of new functionality and processes
- Increase accessibility of platform/application metrics for analysis

Enable Scalable business

- More customers and increased transactions

Architectural Principles

Eliminate accidental design/ code complexity

- Aggressively retire and replace unnecessary complex code and process

Services with single responsibility principle

- High cohesion, low coupling throughout stack

Consistent interface and data flows

- No surprises
- Extensible

Reduce inertia

- Make choices that flavour rapid feedback and change

Design and Delivery Practices

Encapsulate legacy and create seams/interfaces

Java/JS/Golang assessed per service(architect council led)

Utilize REST (JSON/HTTP)

RabbitMQ messaging for async communication

Consolidate, fix & cleanse data per service created

Continuous delivery for all

Automate testing

Better align business goals with development

- Everybody, all together, from early on
- Identity KPI's/success metrics

Technical Leadership

Technical leadership (architecture) skills within the team(s) are vital for building Microservices. Clear business goals combined with technical leadership will help you determine the boundaries of your Microservices and system interfaces.

DevOps

Clearly defined roles and responsibilities for your DevOps team are critical when working with Microservices. Every DevOps resource should be easily mappable in a RACI matrix (Responsibility, Accountable, Consulted and Informed).

DevOps should focus on the areas that matter: Continuous Integration/ Deployment, Logging, Monitoring, Change Management, Incident Resolution etc.

Tools

When choosing appropriate software tools/tech stack, ensure you are choosing tools to support your goals and approach (not the other way around). If you reach a deadlock on one or more equally plausible tools, use the "Spine" model to help you decide. Moving up the spine helps you identify what you really value as an organization which in turn helps narrow the possible toolset.

Feedback

If your Microservices don't collect feedback data, then you will not be able to address failings or take action to adapt.

Your Microservices should be built from the ground up with metrics and alerts at the business (e.g. dashboards), architecture/dev (e.g. code analysis tools) and operations levels (e.g. logging, monitoring). This is vital in order to test the effectiveness and value of your services.

Benefits of Microservices when done right

Ownership of service – Each service built and operated by cross-functional teams which reduce issues and risks.

Services being independently built-deployed-tested , it **speeds up development and release.**

Adaptive Scalability – Scaling up the services which receive heavy load can be done independently rather than scaling the entire application.

Choice of technology – Allows to use the right language, framework independent of the services which fulfill the required need.

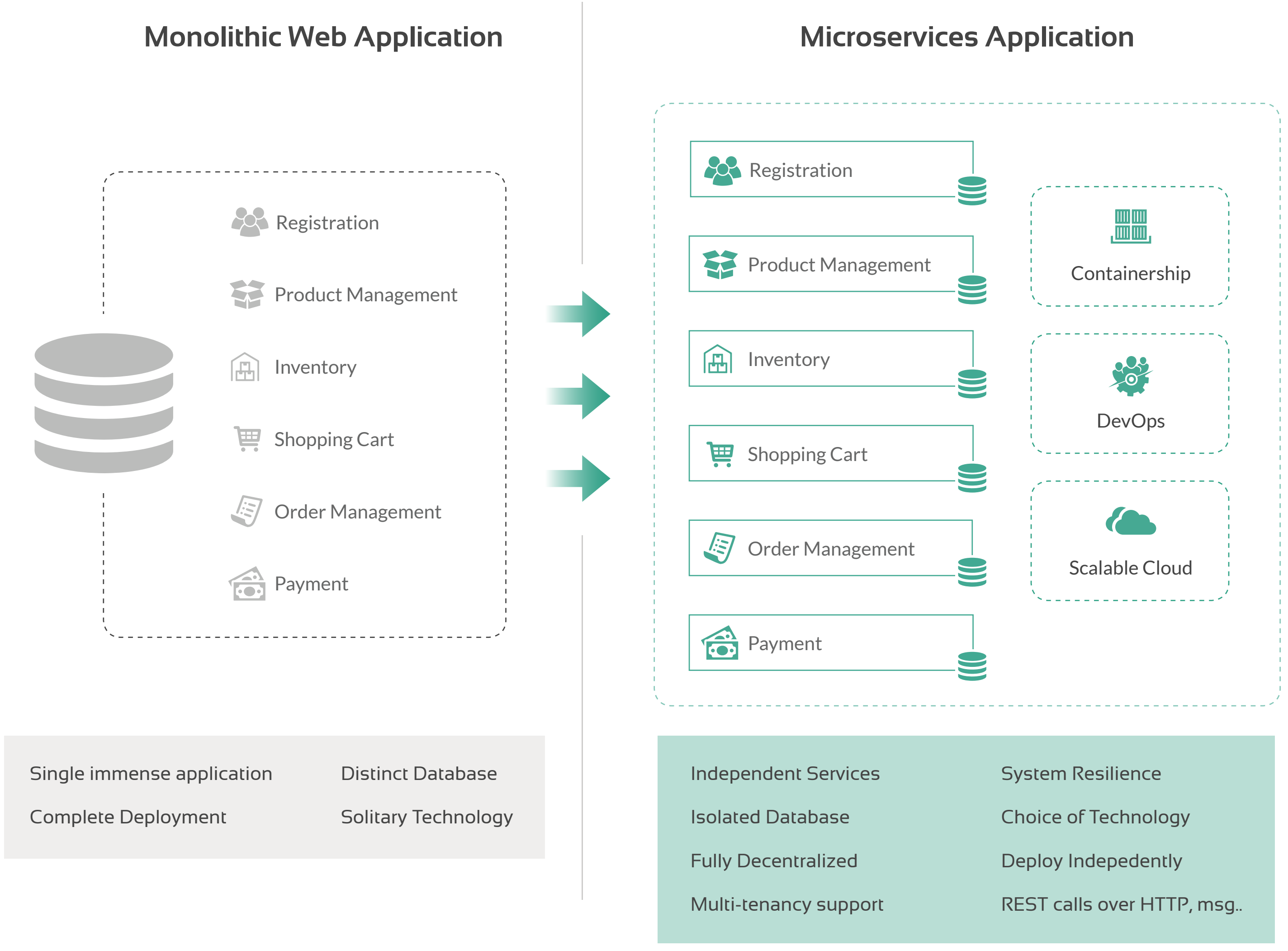
Zero Downtime Deployment - achieved by continuous deployment without interrupting the operation of the services.

Incremental innovation – Businesses see new products and innovation comes through faster release of the application.

Microservices allow businesses to move way faster and smarter with innovation at the core of the product development through plug & play backend services.

Microservices approach results in far more efficient use of code and underlying infrastructure which reports significant cost savings of 50% reduction in infrastructure use.

Let us see how a typical e-commerce application is transformed into a Microservices based architecture. The big monolithic web application is separated into different small services that act independently with their own specific database.



Common approaches for refactoring a monolithic application to Microservices

Approach 1

Replacing existing modules within the monolithic web application into standalone Microservices and repeating this would shrink down the entire monolithic into Microservices – Journey to transform monolithic into Microservices to accelerate digital transformation and evolve at pace with constantly changing the world

Approach 2

Developing new features/modules as Microservices with an API for each of them and interacting with the Monolithic API's – Incorporating product innovation using new technology above old system without changing the core functions

Approach 3

Developing a new Microservices based application from scratch – Greenfield approach bringing out efficient application aligning with the envisioned goals with the knowledge of the scope of services and boundaries

Each approach has its own benefits, risks and technical challenges.

The 4 stages in refactoring monolithic code to Microservices

Identifying new packaging (services) practices by revisiting the monolithic packaging structure

Splitting up Front-end & Back-end into components and extracting each service revised with granular levels and package them independently

Refactoring the data structures that the applications are built on

Finally, the Refactored application is deployed inside containers (or) virtual machines from Iaas providers like AWS (or) using OSGI (Open Service Gateway Initiative)

The User Interface Play

With refactoring the Back-end into several services, the front-end is componentized by web components where it acts as atomic UI piece which is composable and reusable and works with other pieces independently.

Trust In Testing Microservices

Testing of Microservices is the first step for making the service reliable for enterprises and user. With the appropriate testing strategies applied across independent team within Agile/ DevOps lifecycle will result in product quality confidence.

Node.js and Microservices

Node.js has become the go-to technology in the past years for enterprises who are determined to embrace Microservices. The combination of these two technologies can improve application start-up times, general responsiveness and reduce memory footprint.

NETFLIX

Reduced its 40 mins Startup Time with Node

GoDaddy

Switched to Node.js from .Net

Our Experience

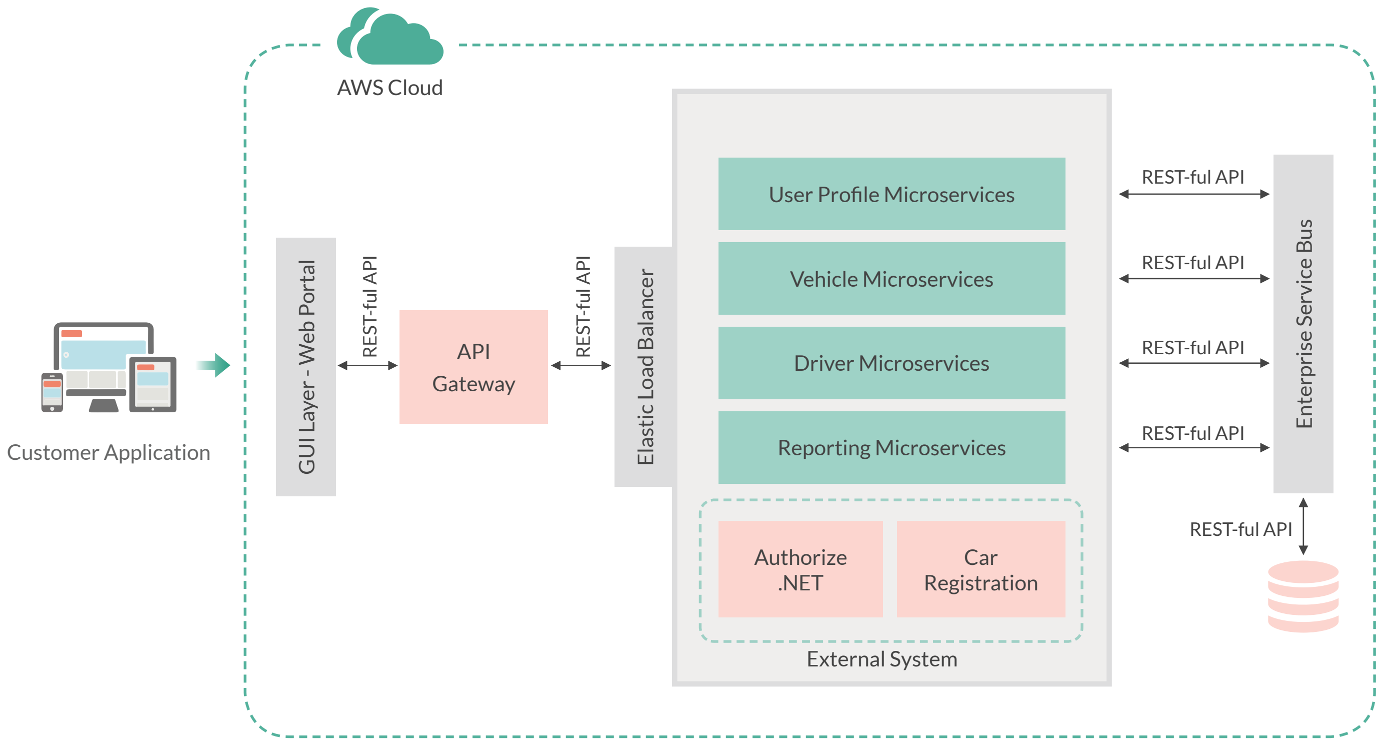
Cloud-hosted Fleet Management Product using SaaS model for small to medium fleet operators

Business Case

A UK based company who are a market leader in fleet management software engaged with Changepond to develop a fleet management application for small to medium fleet operators. The customer's long-term roadmap for the product would require that certain functional entities within the system be implemented as Microservices

Changepond Solution

Changepond worked with the customer's stakeholders to identify and elicit the business requirements for each functional area that would benefit from being implemented as a micro service. These areas were then developed as Microservices and integrated via web services. The resulting solution was deployed on the AWS platform



Key highlights

UI Layer

Solution User Interface was built using Angular and Bootstrap. Each interaction in the portal is processed using API gateway via REST-ful APIs.

UI as a Microservice

A consistent, responsive and user-friendly UI across multiple devices (desktop, mobile, and tablet) was a key requirement. Apart from the web-based portal, Changepond also developed a mobile application that would be used by drivers while they are on the road.

Independent Microservices

Core database entities such as vehicles, drivers, users as well as reporting functionality were developed as independent micro services. They were developed in Java and integrated through REST-ful API (via token based authentication). Each Microservice was deployed separately on AWS EC2 instance. From a technical perspective, this shortens the release lifecycle and also allowed for easier maintainability of the application.

Vehicles as a Microservice

Customer envisioned the product

being positioned as a transactional platform and/or used vehicle marketplace in the long run. A particular fleet operator will post a requirement for a new vehicle within the system and manufacturers/ suppliers can consume the requirements via the vehicle service in order to bid for the sale. Similarly, 3rd party used car sales platforms can consume the details of cars that are available for sale.

Reporting as a Microservice

The new product has been architected on a tenant based model – i.e. each fleet operator has his own database. By implementing the reporting functionality as a Microservice, each operator can consume the reports relevant to their BI needs.

API Gateway

The application utilizes Amazon's API Gateway as an enterprise services bus for faster data processing and improved application performance. This enables customer to integrate with the following external services:

CarWeb: CarWeb has one of the most comprehensive databases on UK registered vehicles. If a particular registration/VIN number is available in CarWeb, the application can automatically pull the car's details directly from CarWeb's database instead of having to manually enter the details in the application.

Authorize.NET: The application integrates with Authorize.NET as a 3rd party multi-currency payment gateway.

Amazon Elastic ECS load Balancer is deployed before the services to sustain scalable of the application during peak hours.

Data Layer

Each core Microservices is connected independently to the Database via Enterprise Service Bus (ESB). The ESB is introduced with future enhancements in mind so that the application could also be integrated with other enterprises systems and databases.

Solution Benefits

Enhanced application scalability and improved system performance by over 800%

Reduced infrastructure footprint by 50%

DevOps practices to improve build and deployment process

ABOUT CHANGEPOND

CHANGEPOND Technologies is a “Native Digital Integrator”, since 2000 helping customers manage the convergence of enterprise software, digital and data solutions.

www.changepond.com

New Jersey | London | Basel | Chennai | Kuala Lumpur