

How to your Convert Monolithic to Microservices

Karthikeyan VK

@karthik3030 

<https://blogs.karthikeyanvk.in>



Why Microservices ?

- Resilience
- Scaling
- Removing Rewriting barrier
- Ease of Deployment



Why Microservices ?

- Organization Alignment
- Composability
- Technology Heterogeneity



Organization Alignment

- Microservices allows to align our architecture our organization, by helping us minimizing the number of people working on one codebase
- Smaller codebases tend to be more productive.
- Support Devops

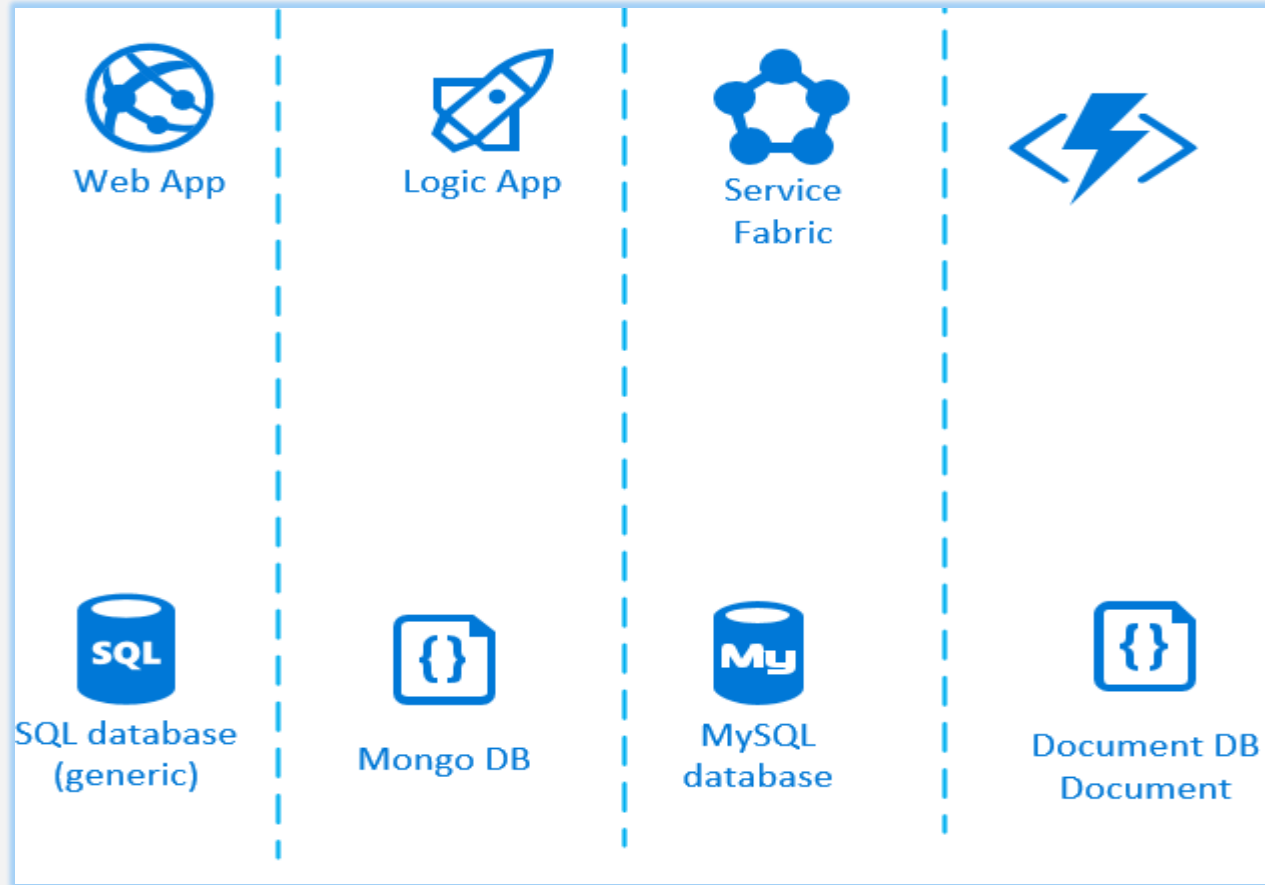


Composability

- Removing thinking in terms of narrow channel to more holistic concepts of customer engagement such as web/native/mobile/tablet/Wearable device and exposing API as service



Technology Heterogeneity



What are Microservices ?

Microservices are small, autonomous services that work together.



How Small is Small ?

- Lines of Code ?
- Who has a system that is too big and that you'd like to break down?



How to define being small ?



- Smaller the service, the more you maximize the benefits and downsides of Microservices architecture.
- When you get smaller, the benefits around interdependence increase
- When you get too smaller, the complexity emerges from having more and more moving parts



4 Challenges

 **redhat.**
Top 4 challenges when implementing microservices:

Corporate culture

Microservices management

Diagnostics & monitoring

Time & resources

Source: Red Hat 2017 Microservices Survey.
Conducted by TechValidate, Nov. 2017.



Corporate Culture

- Open to technology
- Think of Task Force
- Leave behind process and procedures
- Culture of Automation
- Expect failure
- Autonomous



Microservices Management

- Defining Strategy
- Implementing new projects
- Migrating existing project



How to start ?



Pet Store Functionality

- Authentication
- Search Pet
- Order Pet



What Makes a Good Service?



Main Principles in Splitting Microservices

- Loose Coupling
- High Cohesion



Loose Coupling

- Loosely coupled service knows as little as it needs to about the services with which it collaborates.



High Cohesion

- Related behaviour to sit together, and unrelated behaviour to sit elsewhere.
- Changing behaviour in lots of places to deliver a change, signals lack of high cohesion



Practices

- Practices should underpin our principles
- Sometimes practices reflect constraints in your organization
- Coding guidelines
- Integration Style – HTTP/REST



Combining Principles & Practices

- One person's principles are another's practices
- .NET Team practices and Java Team practises based on principles



What is Bounded Context?

- Specific responsibility enforced by explicit boundaries.



What is Bounded Context?

- A Domain consists of multiple bounded contexts, and residing within each are models that do not need to be communicated outside as well as things that are shared externally with other bounded contexts.

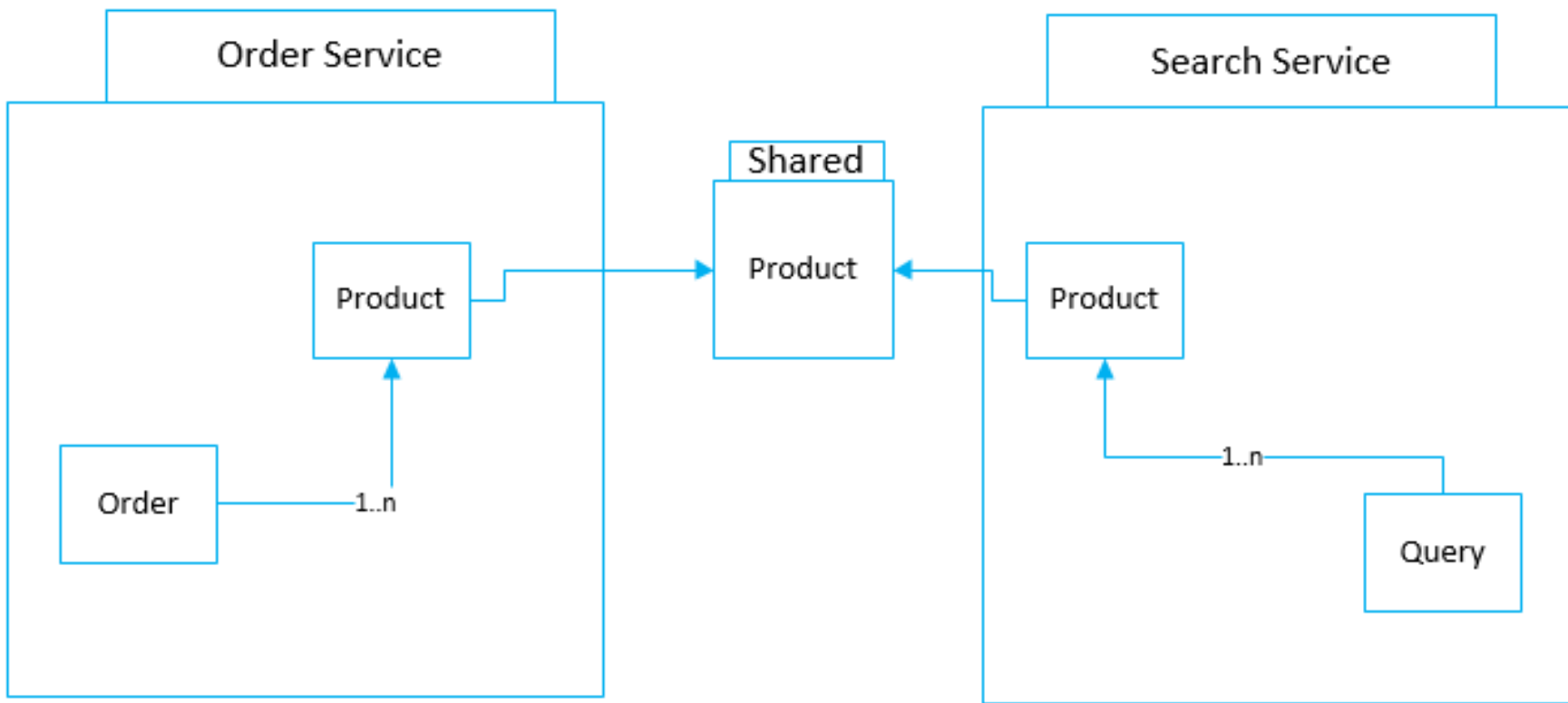


What is Bounded Context?

- Each bounded context has an explicit interface, where it decides what models to share with other contexts.



Models & Shared Models



Modules and Services

- What models should be shared, and not shared
- Identifying boundary within our domain.
- Keep related code together and attempt to reduce the coupling to other modules in the system.
- Microservices should cleanly align to bounded contexts.
- Once proficient, skip the step of keeping the bounded context modelled as a module. Jump straight for a separate service



Shared Libraries

- Monolithic codebase into multiple libraries.
- Libraries give you a way to share functionality between teams and services.
- Too much leads You to lose true Technology heterogeneity.
- Solution is to create common tasks that aren't specific to your business domain that you want to reuse across the organization



Premature Decomposition

- Don't start decomposition too quickly.
- Initial Service boundary will not be right
- CI/CD will break
- Decompose into Microservices is much easier than trying to go to Microservices from the beginning.



Business Capabilities

- Don't think in terms of data that is shared
- Think in terms of bounded context capabilities that is provided in the rest of the domain.
- Communication in Terms of Business Concepts



Integration between Microservices

- Avoid Breaking Changes
- Keep Your APIs Technology-Agnostic
- Make Your Service Simple for Consumers
- Hide Internal Implementation Detail



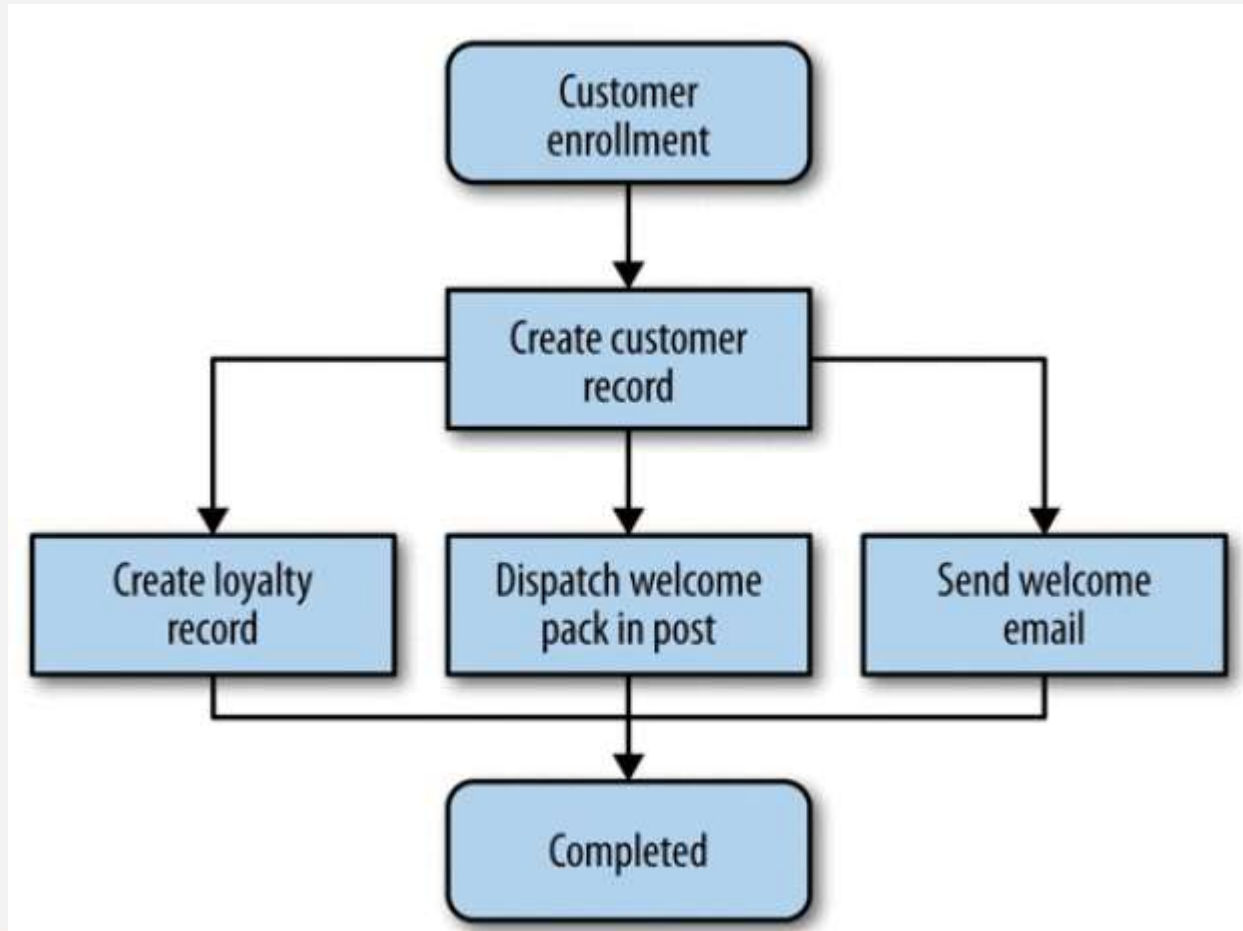
Interfacing with Customers

- The Shared Database



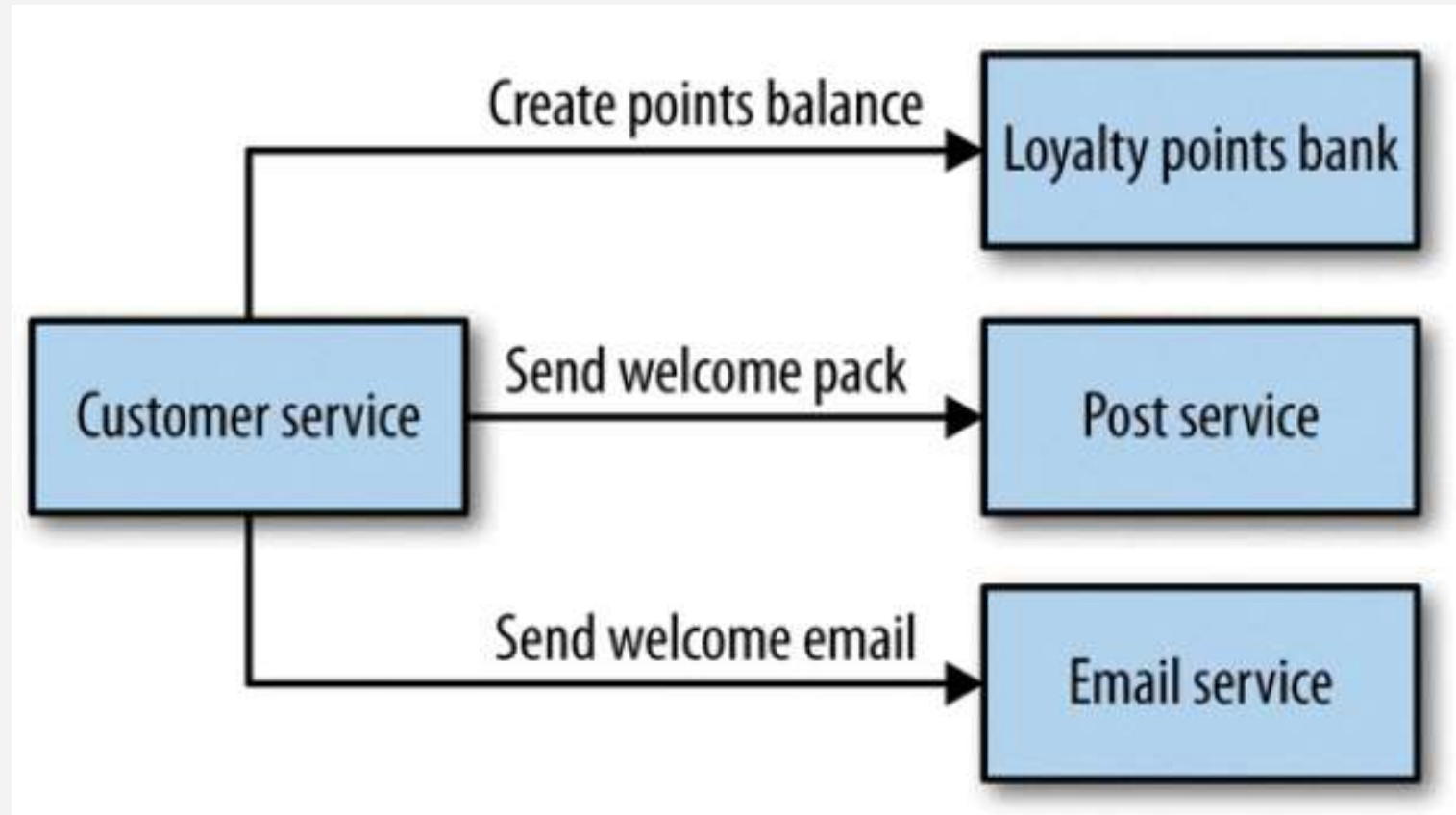
Orchestration Versus Choreography

- Orchestration



Orchestration Versus Choreography

- Choreography



Conway's law and System Design

- Any organization that designs a system (defined more broadly here than just information systems) will inevitably produce a design whose structure is a copy of the organization's communication structure.



References

- <https://middlewareblog.redhat.com/2017/12/05/the-state-of-microservices-survey-2017-eight-trends-you-need-to-know/>



Thank you /Q&A

