Microservices: Brownfield: Introduction

acarlstein.com/

Posted by Alejandro G. Carlstein Ramos Mejia on January 11, 2017 January 18, 2017 About Programming / Architecture / Microservices

Microservices: Technology | Microservices: Brownfield: Migration

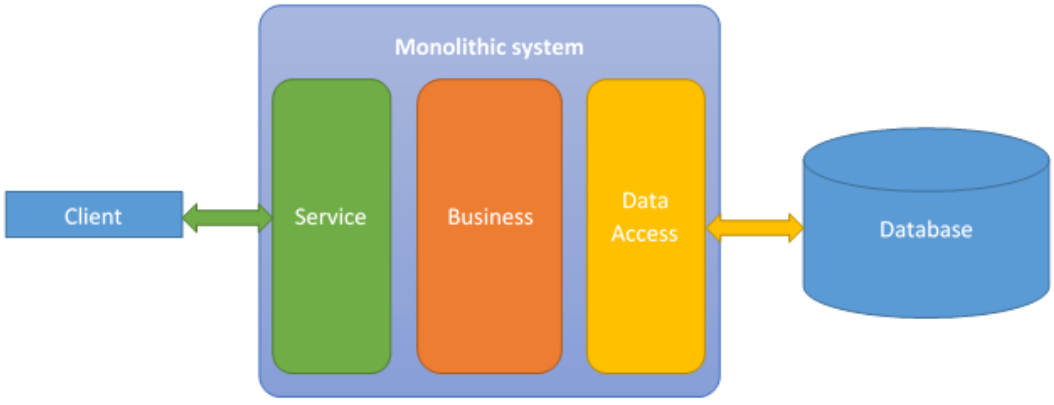
The term Brownfield (used in IT) means that we are going to do development and deployment of new software systems while taking into account previous existing legacy software systems. The term Brownfield originates from the contemporary civil engineering where in some cases required that new buildings were designed and erected while having in consideration other structures and services already in place. In our case, transforming a monolithic system into a micro-service system can be considered Brownfield.

Approach

So we have an existing monolithic architecture and needs to be migrated to a micro-services architecture.

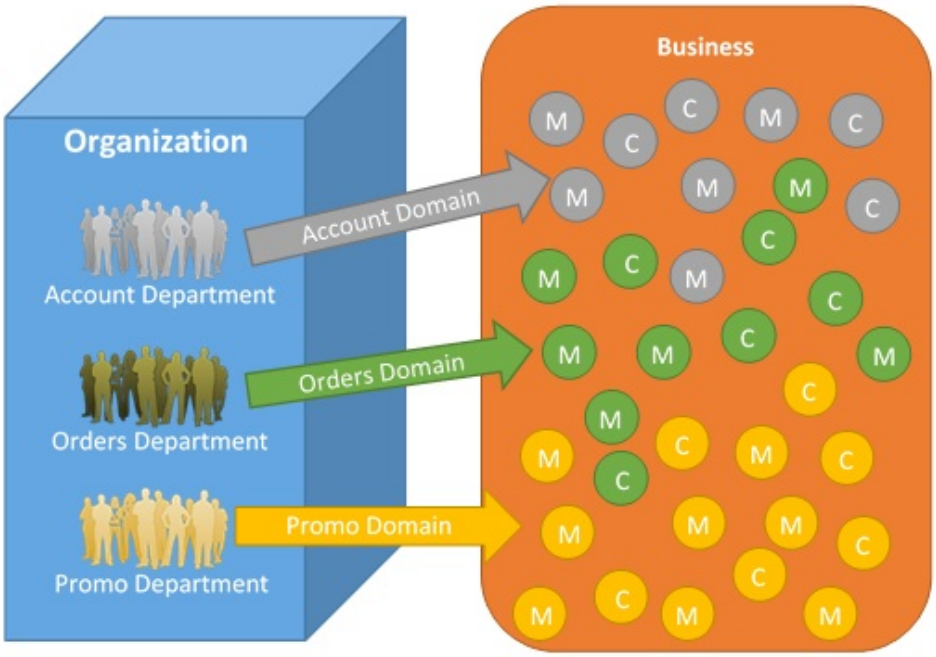
1. Decide on the initial approach to take in order to ready your system for the migration from the monolithic architecture to the micro-services architecture.
2. Take a look to the actual migration about how to migrate a monolithic system to a micro-service system.
3. Our monolithic system may have one large database; therefore, we will need to do a database migration from this large database into multiple databases to support our micro-services.
4. Take a look into the effect our micro-services architecture may have over the transactions within your system and reporting within your system.

Our existing system is a monolithic architecture that most likely have grow to a huge size. The most likely, the business section of our monolithic architecture is large and complex.



We can assume that our current system lacks most of the micro-services design principles; therefore, the first step is to figure out how can we begin introducing those principles.

Let's begin by analyzing the code of our system and try to identify the seams that form part of it. We need to do an identification and separation that reflects the business domains within our business. We may be able to identify modules, classes, and methods that are related with specific parts of our organization so we can group them based on the business domain.



Just for your information, the separation doesn't have to be exactly based on the domains of your organization, it can be also done based on the functions within your organization.

You may find code that overlaps from one department to another. In this cases, we must refactor the code so it can be split into the bounded context.

Begin by creating a module for each bounded context, then begin moving the code around in increments. Then, ensure to validate your code refactoring by implementing unit tests and integration tests that validate your changes.

After every release, review the code again, then refactor again, and keep repeating this process until you can have bounded context clearly defined.

When you finish having well define bounded contexts then you are going to be able to create micro-services boundaries. This means that we are going to transform these bounded contexts into micro-services.

© 2017, Alejandro G. Carlstein Ramos Mejia. All rights reserved.