MDM Microservices

Gurpreet Singh *
Infosphere MDM Delivery, IBM

June 18, 2017

Abstract

A proposal to migrate existing tools and develop new tools using a microservice based architecture. A microservice based approach provides solutions to problems regarding property management, user friendliness, and many internal development issues. A proof of concept architecture implementation is attached and documented to show the potential benefits to the team.

Keywords: Bluemix, Docker, Golang, Kotlin, Javascript, rMarkdown

*The author gratefully acknowledges ... David Song, Simon Kotwicz, and Herman Singh

1 Problem Analysis

1.1 High Level Goals

My interpertation of the contest requirements are summerized into the following 3 points

- More user friendly tooling
- Easier for tools to access common properties
- Elegent solution for property management

1.2 Unrealized Problems

The following is a list of problems I realized throughout the past year using the current set of tools.

- Tools unable to communicate with each other
- Incredibly difficult and complex learning curve to reach 1 successful run of any tool
  - Too many tutorials to follow, too many settings to change just to run a simple test
- Language and platform restrictions becoming larger with each new tool
  - Restricting possible approaches to a new problem
- High overhead during initial setup phase for every tool
  - Only a few people know how to set everything up
  - Leads to too many environment problems that should never occur for a user
- Current programming paterns make it hard to implement test driven development

2 Solution

2.0.1 Microservices

Microservices architecture is an approach to application development in which a large application is built as a suite of modular services. Each module supports a specific business goal and uses a simple, well-defined interface to communicate with other sets of services. (Matturro 2017)

- Everything is a microservice
  - Tools handle one specific task, very efficently
- Tools are developed and run out of Docker containers
- Tools are setup on central machines and run through an interface (web / CLI / anything)
  - This means one time setup and maintainence, no more setup issues.
- Inter-tool communication is done through HTTP
  - HTTP is easily accessible from any platform and any programming language
  - Using a series of HTTP methods (GET, PUT, POST, DELETE...) anything can use a tool
- Every tool will have a set of API endpoints that clearly define that tool's usage
  - We could have one tool written in Java, another tool written in Pythong and they could both work together without any modification.
  - A clear API will enforce and encourage clear tests
- A set of endpoints will be required for any tool to work with the architecture
  - Each tool will register on bootup with the Core service providing its required property set

  - Provides a /status and /run endpoint for UI interactions
  - Informs Core when a run has finished

- Property values will be stored on the Core service
  - Each tool will use the /fetch endpoint and send in a run id to retireve the properties it needs

3 Functional Requirements

`Let an interface be the WebUI or any other user friendly frontend able to call HTTP methods to a service`

`Let a tool be the Tool or any other delivery tooling that does work and is able to call HTTP methods to a service`

3.1 Core

- Allows new services to register. Stores their properties.
- Allows an interface to fetch list of services
- Can check the status of any service
- Allows an interface to create a new run with any predefined property set
- Queues up requests and load balences them between services if there are multiple instances of a tool running
- Allows a tool to submit a run result
- Allows an interface to fetch a run result

3.2 A Tool

- Registers with the Core at the start of the service and sends it a list of required properties
- Able to provide it's status, even in the middle of a run
- Fetches new properties from Core for every run
- Sends a report to Core after every run

3.3 WebUI

- Gets a list of services from Core
- Creates a run with one of the property maps for one of the services retrieved
- Checks back for the result of the run and displays the report

References

Matturro, B. (2017), 'Microservices'.

URL: *http://searchmicroservices.techtarget.com/definition/microservices*