John Hancock

Services Migration Project

“DEV-Complete” Criteria

Date: 05/09/2018

Here is what we agreed upon…

1. Each micro service will have a repository in JH’s GitLab.
   * Developer’s work on “main” branch today.
     + (In discussion with Ankeet, we agreed to simplify branching and rename it to “develop”). **Completed**
   * Leverage tagging for progression status
     + (suggestions : Dev–Complete, Test-Complete, UAT-Complete, Move-To-Prod)
2. Develop micro-service
   * Developers will convert a TIBCO flow into a Spring Boot based Micro-Service using the agreed upon template
     + The currently shared template is for db calls only.
     + The template needs newer version to baseline calls to external services.
     + Template is work in progress.
     + If there are new changes to template those changes will be applied to services are yet to be developed (not refactoring already created services, unless a critical vulnerability).
     + The service will provide REST endpoints
       - JSON based Request and Response (application/json)
       - The SOAP – JSON wrapper will be implemented in Apigee layer. (Yogesh’s team).
   * Developers will, for each JSON based REST micro-service, create the following
     + Unit tests (junit)
     + Mock tests (junit) as needed (if cannot use H2 or its an ext service)
       - Add tests for Rest endpoints (JSON)
     + Smoke tests (deployed to dev pcf, and run a set of tests)
     + These tests will be run in automated fashion using the CI/CD pipeline.
   * Add exception handling
     + Enhance the template with the agreed upon pattern (JH).
   * Add swagger documentation and Readme.
     + For the rest endpoints of the service
     + Swagger documentation will be disabled on the *cloud* profile, only enabled for *local* profile
     + Swagger documentation generated during build time, add step to ci/cd pipeline to move to central location (Artifactory).
   * Add logging patterns
     + Today, JH has provided a jar file, included in each project, to define logging patterns.
     + Plan is to look for alternatives
       - Spring Sleuth / Zipkin for distributed tracing and set the log message pattern in templates.
       - Log4j2 for masking
       - Override toString() to return Json (sensitive properties will be marked with @JsonIgnore, or @JsonIgnoreProperties)
       - OR continue to use the jar file (needs enhancement for masking)
3. Developer will create the CI/CD pipeline for the developed micro-service. The CI/CD pipeline will be based on the template agreed upon with JH.

The service will do the following

* + Compile, Run automated tests
  + Build Deployable Archive
  + Push to PCF (dev instance)
  + Future enhancements
    - Push archive to Artifactory (JH does not have an artifactory today and would need to stood up on Azure) - Completed
    - Semantic Versioning (need more review JH)
    - Notification (See - <https://concourse-ci.org/community-resources.html>) Need more review from JH

1. Default patterns that developers will implement in a micro service
   * 12 factor principles
   * TDD & CI/CD
   * Config Server
     + Based on Git Repo
     + **Config file will not have any secured credentials.**
     + Require JH to upgrade PCF to latest version – JH.
   * Service Registry
     + A microservice will be build with auto register with service registry with a service instance name of “registry” or “service-registry”.
     + Apigee has been identified as a service registry by JH. JH will do a POC to see if the SCS service registry instance can be replaced with Apigee service registry without additional changes to each service.
   * Circuit Breaker (only applies for Edge Services)
2. JH responsibility
   * Firewall requests for external systems
   * Apigee service registry
   * BU testing

**Need POC**

1. SOAP based Request and Response (application/xml aka xml/http)
   * Validate with POC (<https://howtodoinjava.com/spring/spring-boot/spring-boot-soap-webservice-example/>)
2. Need a POC on masking data for logging.
   * Review logging, that Antony put together

**Outstanding**

1. Translation of reference list
   * JH has an existing utility.
   * Can we convert it into a service that can cache the ref data in to a Redis cache