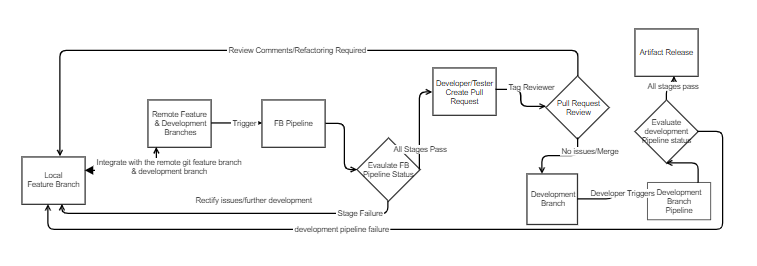
Automated SDLC

The following Automated SDLC is in place for the SFM Team. A gitflow branching mechanism for the microservices pipelines support most of the stages required.



1. Each microservice has a feature and development branch pipeline. All stages must have a successful outcome on both pipelines before an artifact can be released to maven as a released version. Snapshot versions can be released at the end of the feature branch pipeline, and a stage will be added for same. Snasphot versions are important in order to enable dependency services to have an early view of a service.
2. Developers must integrate their local feature branch changes with the latest changes in development and their corresponding feature branch and verify results on a Feature Branch Pipeline before submitting a pull request.
3. The sprint cycle spans two weeks. It is the developer’s responsibility to ensure that pull requests are submitted as early as possible, to allow adequate time for review & refactoring. A Pull requests submitted the day before a sprint ends does not allow enough time for both a review & refactoring should it be required. Therefore, they must be submitted at least by the Wednesday of the week on which s sprint closes. Merge Requests must include a link in the description to the Jenkins Build that reflects the last commit available in the Merge Request. If the last commit on the Jenkins Build does not reflect that on the PR it will be rejected. If a PR requires a subsequent review, an updated jenkins build link needs to be added. The reviewer should mark a PR with “Wait for Author” and tag the author after a review, likewise the author should mark a PR with “Wait for Reviewer” and tag the reviewer once the refactoring is completed.
4. POMs must be at a SNAPSHOT version in feature branches, the pipelines shall gate this. POM versions must be updated to a non-snapshot versions before being released. The pipeline shall have stages to manage validation on same.
5. The pipelines should mange the tagging, and publishing of the images to the Openshift Kubernetes registry.
6. Microservices deployed to an Kubernetes cluster from both feature and development branches will be distinguished by namespace.
7. Integration Tests will be carried out against the deployed microservice on Openshift. Where on-premise systems do not have a test instances available, mocks will be created by Testers.
8. The current implementation is an early release of an automated SDLC for microservices. Planned enhancements include
   1. The automatic seeding of Microservice Jenkins Jenkins Jobs from the SpringBootInitializer
   2. A SpringBoot Ininitializer application to auto create a microservice based on selection, including the addition of same to a git repository
   3. Pull Request Builder Automation
   4. Quality Gates
   5. Automated Deployments to Kubernetes cluster
   6. Continuous Monitoring
   7. API Management
   8. Automated Testing BDD, API Testing, Use of Mocks
   9. Enhanced Tooling to orchestrate the Azure Subscriptions provided by GWLE

**Standard Artifact Release Management is in place for Maven Artefacts**

A change of an artefact for a **particular release**/**Request for change**, denotes a new required version.  i.e 1.0.0 becomes 1.0.1-SNAPSHOT

 If that artefact is not completed for the release the version should not be incremented again to 1.0.2-SNAPSHOT, should the development fall into the next sprint.

 Once a snapshot version is tested, and verified on a feature branch, and then on a development branch, it then becomes a release candidate and in this example it would be promoted to 1.0.1

 Semantic versioning is used, Major.Minor.Patch until these artefacts are promoted to the live environment they will not be considered for Minor or Major releases.