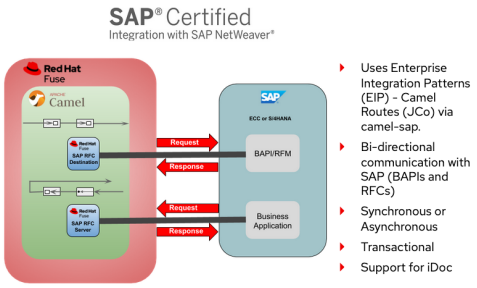
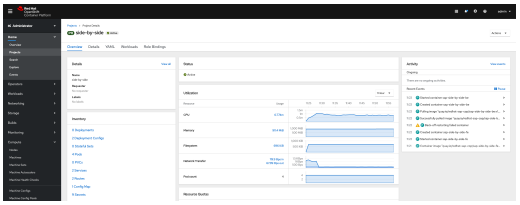
Exercise 2: Using Fuse to create Camel routes with S/4HANA via JCo and OData, exposing these as REST API endpoints

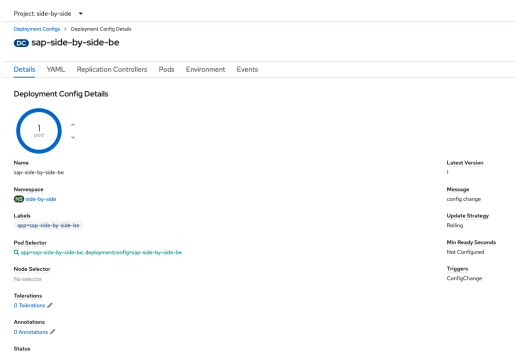
This scenario will show how to integrate with SAP S/4HANA from a microservice running in OpenShift using a **certified integration** by SAP. To do that this workshop provides a running backend microservice based on this repository. This will use a Spring Boot Camel environment to invoke remote function modules and BAPI methods within SAP.



In addition, these routes are exposed as REST API endpoints in our backend microservice and presented via Swagger UI for testing and validation. Click on Home -- Projects -- side-by-side to get an overview of the Project already present in your OpenShift Cluster for this workshop.



Let's identify the backend microservice and get the external exposed URL for it. In the Project Overview, you can see the Inventory section on the bottom left. From that section click on Deployment Configs. You can get to the same place using the Workloads -- Deployment Configs menu. Once you get there, you will see 2 different Deployment Configs. The one associated with the backend microservice is sap-side-by-side-be. If you click on it, you will be able to see more details like the number of Pods running for this Deployment Config or the container image used which is quay.io/redhat-sap-cop/sap-side-by-side-be:v1.1.0 by the time these instructions where written.

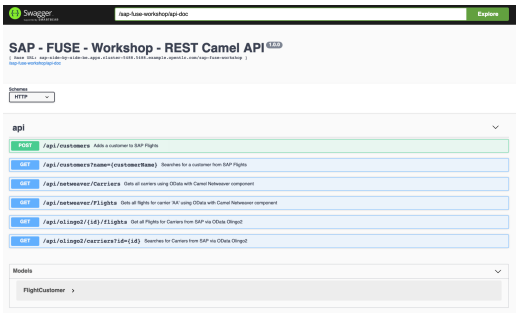


You can review more details from this Pod, check the actually used resources or check the logs coming from the Pod to see if everything looks fine, so you get familiar with it, as you can come

back later to the Pod and see these values again once you start hitting the exposed API endpoints in the microservice.

Let's review now how to access the exposed URL for this microservice. As explained before, Swagger UI has been configured so the documentation and testing for each API method are available for quick tests and to be used as a reference for developers to consume this API. Click on Networking -- Routes menu and you will see 2 different routes available. The one specific for the backend microservice is called sap-side-by-side-be. You can click on the exposed URL referred in the Location column.

Once you click in the URL link this will be opened in your default browser and you will get a **Whitelabel Error Page** error. This is expected as you need to append the following path to the URL to get to Swagger UI page: /webjars/swagger-ui/3.28.0/index.html?url=/sap-fuse-workshop/api-doc#. Once you add this you should be able to see Swagger UI for this specific API with all the available methods so you can test those and get more information from each one.



As an example, check how we can retrieve users from SAP HANA for the SFLIGHT demo schema:

[Swagger 2]

Try to test a couple of more methods from Swagger UI and check the logs from the backend Pod at the same time, so you can see the correspondence:

[Swagger 3]

With all this together we have reviewed the integration capabilities Red Hat Integration can bring to SAP customers, to allow them to integrate their custom SAP extensions and code into S/4HANA. If you want to deeply review how all this work together check the backend microservice repository where you can find all the code that makes this work.