



Tech Talk

Streamline Your Architecture with Red Hat Camel and AMQ Streams

Red Hat Team

Date 12-Aug-2024

Table of Contents

1 Delete the previous workspace from the Devspace	4
2 Import the DeveloperHub template for this demo.	4
2.1. Go to the Create Software Template section	5
2.2. Register the new template	5
2.3. Finish the import	6
3 Create the Catalog from template	6
3.1 Go to the Template Create section	6
3.2. Choose the template "Create a Camel app for OpenShift"	7
3.3 Add the details, Name and Cluster Domain, copy from the browser url as shown below	7
3.4 Click Next and verify the repo details and create	8
3.5. The template pipeline will be executed for the initial setup	8
3.6 Verify the Catalog	9
4 Open the new workspace from the developer Hub.	10
4.1 After waiting a few minutes for OpenShift Dev Spaces to finish setting up your workspace, You click the button Yes, I trust the authors.	10
4.2. Verify all the extensions have been installed correctly.	12
5 Module 1	12
5.1 Update the application.properties	12
5.2 Commit the Changes	14
5.2.1. You click on the Source Control icon located in the left menu.	14
5.2.2. Then, you enter the commit message "updated application properties" and click on the Commit button to finalize your changes.	14
5.2.3. In the pop-up window that follows, you click Yes to stage your changes.	14
5.2.4. Finally, you click on the Sync Changes button.	15
6.2.5. In the pop-up that follows, you click OK to push your changes and complete the process.	15
5.3 Verify the pipeline	16
5.4 Open the application	17
6 Module 2	17
6.1 Update Expression of SetBody	17
6.1.1. Go to src/main/java/resource/Module1.camel.yaml	17
6.2.2. Update the setBody component with below expression	17
6.2 Commit the Changes	18
6.3 Verify the pipeline has been completed successfully.	19
6.4 Verify the Output	19
7. Module 3	19
7.1. Click on the 3 dot of the log component and select append	19
7.2. Select the xj component from the catalog	20
7.2.1 Update the below configuration for the xj component	20
7.3 Commit the Changes	22

7.4 Verify the pipeline has been completed successfully.	22
7.5 Verify the Output	22
8 Module 4	23
8.1. Click on the 3 dot of the XJ component and select append	23
8.2. Select the kafka component from the catalog	24
8.2.1 Update properties for the Kafka component and close	24
8.3 Commit the Changes	25
8.4 Verify the pipeline has been completed successfully.	25
8.5 Verify the Output	25
9 Module 5	26
9.1. Create a new route	26
9.2. Replace Timer with Kafka component.	26
9.3. Select the kafka component from the catalog	27
9.3.1 Update properties for the Kafka component and close	28
9.4. Replace Log with AzureBlob Storage component.	28
9.5. Select the AzureBlob Storage component from the catalog	29
9.5.1 Update parameters for the azure-storage-blob component	30
9.6. Select both the route	32
9.7 Commit the Changes	32
9.8 Verify the pipeline has been completed successfully.	32
9.9 Verify the Output	32
10. Update the DevSpace plugin	33
10.1. Login to the openshift console using admin credentials.	33
10.2. Go to the Installed OperatorPage and look for Red Hat OpenShift Dev Spaces	33
10.3 Update the devspace cheCluster Yaml configurations.	34
10.4. Update the pluginRegistry configurations.	34
10.5. Save the configuration and verify the changes after reload.	35

1 Delete the previous workspace from the Devspace

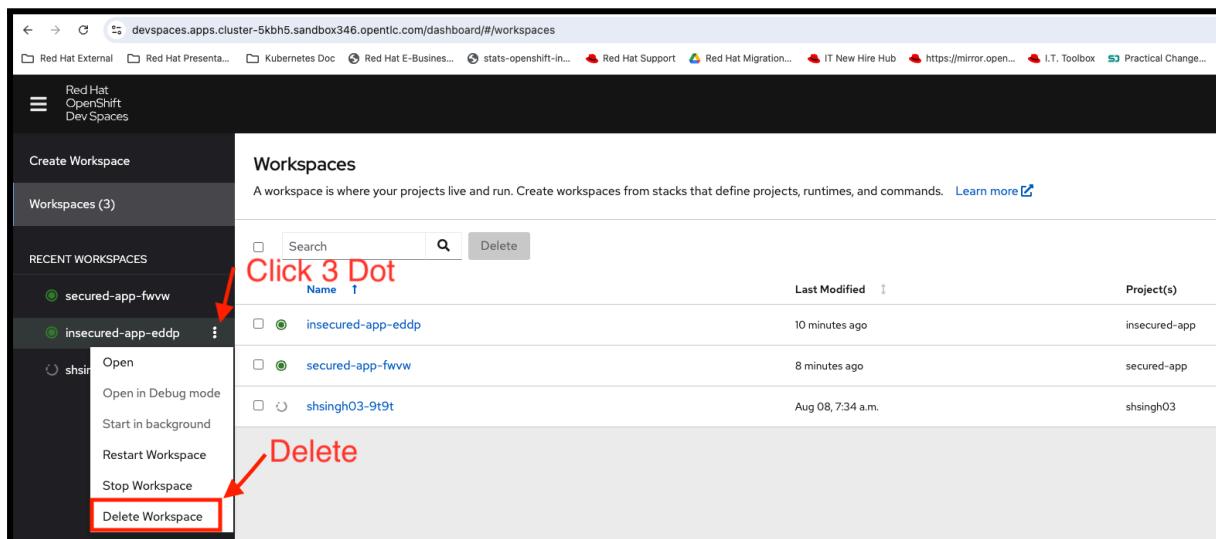
(Only when you have 2 Workspace)

Each of the following sections provides a brief description and a list of recommended actions.
Additional explanations are provided where additional clarity is required.

URL:- <https://devspaces.<CLUSTER-DOMAINNAME>>

SAMPLE URL :-

<https://devspaces.apps.cluster-5kbh5.sandbox346.opentlc.com/>



The screenshot shows the Red Hat OpenShift Dev Spaces dashboard. On the left, there's a sidebar with 'Create Workspace' and 'Workspaces (3)'. Under 'RECENT WORKSPACES', there are three entries: 'secured-app-fwww', 'insecured-app-eddp', and 'shsingh03-9t9t'. The 'insecured-app-eddp' entry has a red arrow pointing to its three-dot menu icon. Another red arrow points to the 'Delete' button in a modal window titled 'Delete' that appears over the workspace list. The main area is titled 'Workspaces' with a sub-instruction: 'A workspace is where your projects live and run. Create workspaces from stacks that define projects, runtimes, and commands.' It includes a search bar, a 'Delete' button, and a table listing workspaces by name, last modified time, and project(s).

Name	Last Modified	Project(s)
insecured-app-eddp	10 minutes ago	insecured-app
secured-app-fwww	8 minutes ago	secured-app
shsingh03-9t9t	Aug 08, 7:34 a.m.	shsingh03

2 Import the DeveloperHub template for this demo .



2.1. Go to the Create Software Template section

The screenshot shows the Red Hat Developer Hub interface. On the left, a sidebar has a red box around the '+ Create...' button under the 'Catalog' section. A red arrow points to this button. The main area is titled 'Software Templates' with the sub-instruction 'Click Here'. It features a search bar, sections for 'Available Templates' and 'Templates', and filters for 'Personal', 'My Org', 'Categories', and 'Tags'.

2.2. Register the new template

Template URL:-

<https://github.com/shailendra14k/backstage-template/blob/main/backstage-camel/template.yaml>

Register an existing component

Start tracking your component in Red Hat Developer Hub

1 Select URL

URL *
https://github.com/shailendra14k/backstage-template/blob/n

Enter the full path to your entity file to start tracking your component.

Analyze

2 Import Actions
Optional

3 Review

4 Finish

Click Here

Register an existing component

Enter the URL to your source code repository to add it to Red Hat Developer Hub.

Link to an existing entity file

Example: https://github.com/backstage/backstage/blob/master/catalog-info.yaml

The wizard analyzes the file, previews the entities, and adds them to the Red Hat Developer Hub catalog.

Learn more about the Software Catalog →

2.3. Finish the import

3 Create the Catalog from template

3.1 Go to the Template Create section

Register an existing component

Start tracking your component in Red Hat Developer Hub

1 Select URL

2 Select Locations
Discovered Locations: 1

3 Review

4 Finish

The file was tracked for the following locations:
https://github.com/shailendra14k/backstage-template/blob/n
Entities: 2
location:generated-7a843d349039a46b49276974835b17
template:spring-boot-camel-on-openshift-template

Click Here

Register another

Register an existing component

Enter the URL to your source code repository to add it to Red Hat Developer Hub.

Link to an existing entity file

Example: https://github.com/backstage/backstage/blob/master/catalog-info.yaml

The wizard analyzes the file, previews the entities, and adds them to the Red Hat Developer Hub catalog.

Learn more about the Software Catalog →

3.2. Choose the template “Create a Camel app for OpenShift”

Select this template

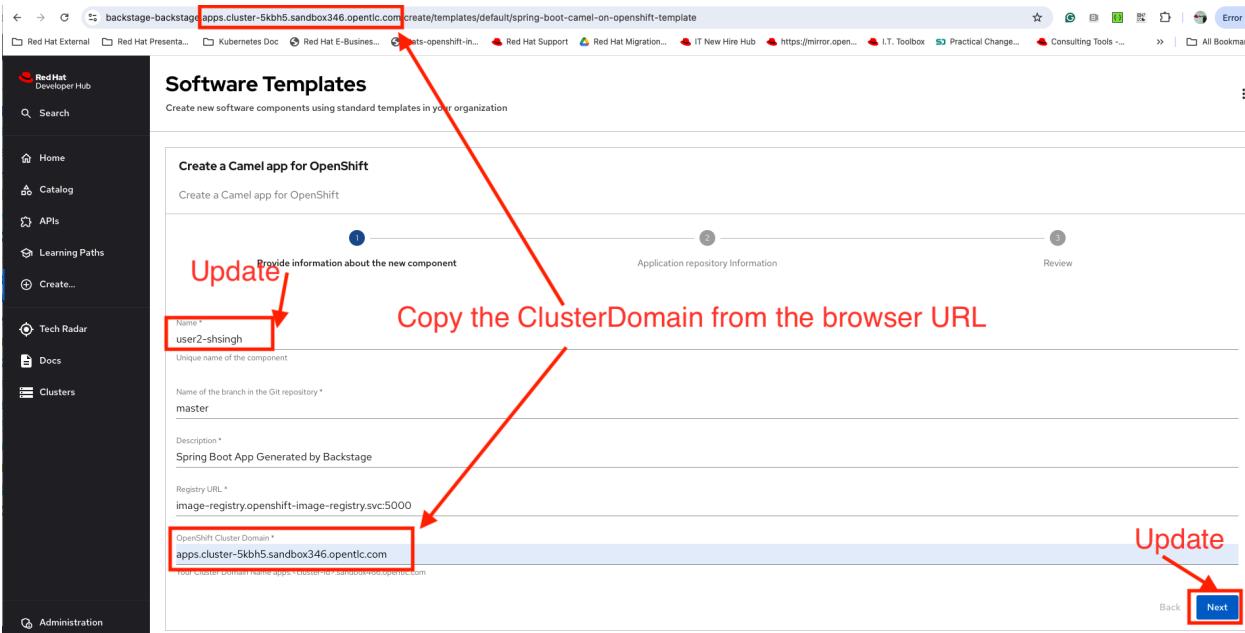
Available Templates

Templates

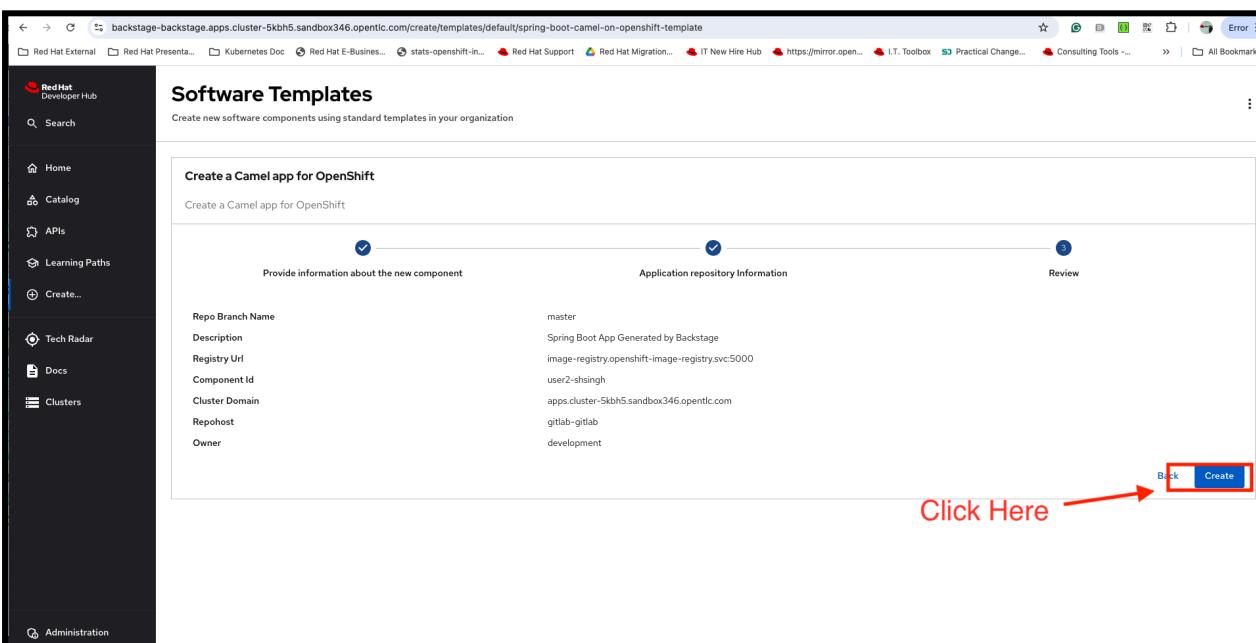
Service	Description	Tags	Created By
Secured Quarkus Service Software Supply Chain	Create a simple microservice built on a trusted software supply chain	recommended, java, quarkus, maven	rhdh
Insecured Quarkus Service Software Supply Chain	Create a simple microservice with no supply chain security	recommended, java, quarkus, maven	rhdh
Create a Camel app for OpenShift		spring-boot, java, maven, tekton, openshift, argocd	shsingh

3.3 Add the details, Name and Cluster Domain, copy from the browser url as shown below

Name :- user1-<yourname> (no cpas)



3.4 Click Next and verify the repo details and create



3.5. The template pipeline will be executed for the initial setup

Run of spring-boot-camel-on-openshift-template

Task d859ee7e-23ca-494c-8c60-f1f8c235c918

Generating the Source Code Component: 2 seconds

Publishing to the Source Code Repository: 3 seconds

Registering the Catalog Info Component: 1 second

Generating the Config Code Component: 2 seconds

Publishing to the Config Code Repository: 2 seconds

Create ArgoCD Resources: 3 seconds

Click Here → **Open the Catalog Info Component**

3.6 Verify the Catalog

COMPONENT - SERVICE

user2-shsingh ☆

Owner: user1 | Lifecycle: demo

Overview Topology Issues Pull/Merge Requests CI CD Kubernetes API Dependencies Docs

This entity has relations to other entities, which can't be found in the catalog.
Entities not found are: system:default/workshop-camel-demo

Links

- OpenShift Dev Spaces (VS Code)
- OpenShift Dev Spaces (JetBrains IntelliJ)

Merge requests statistics

Avg Time Until Merge: Never
Merged To Total Ratio: 0%
Number of MRs: 20

About

View Source View TechDocs

Description: No description

Owner: user1 | System: workshop-camel-demo | Type: service

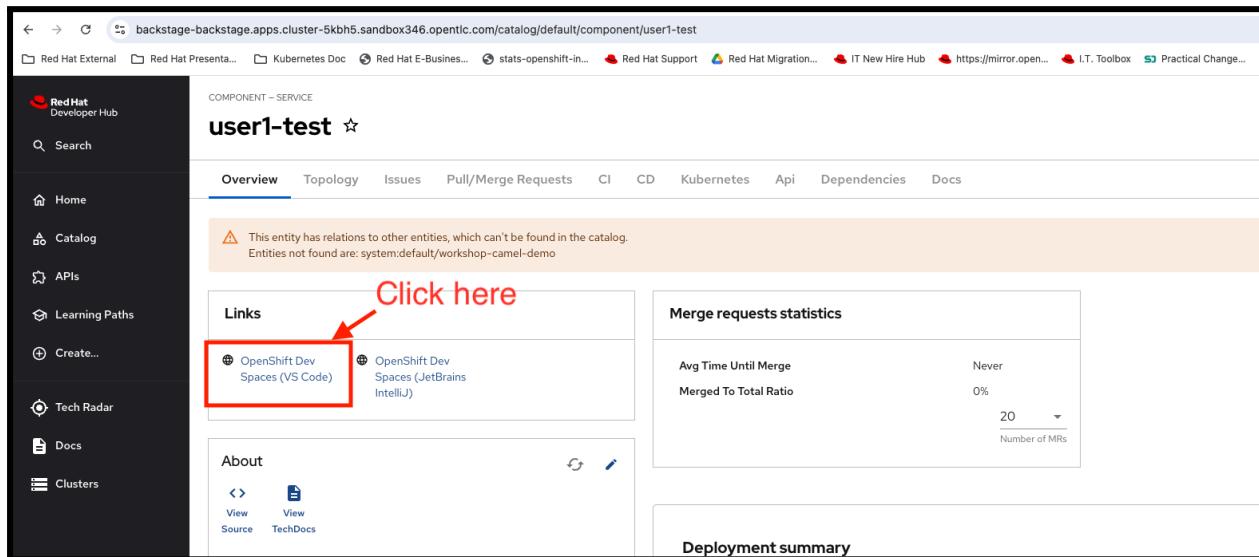
Lifecycle: demo | Tags: spring-boot, java, maven, tekton, argocd, renovate

Deployment summary

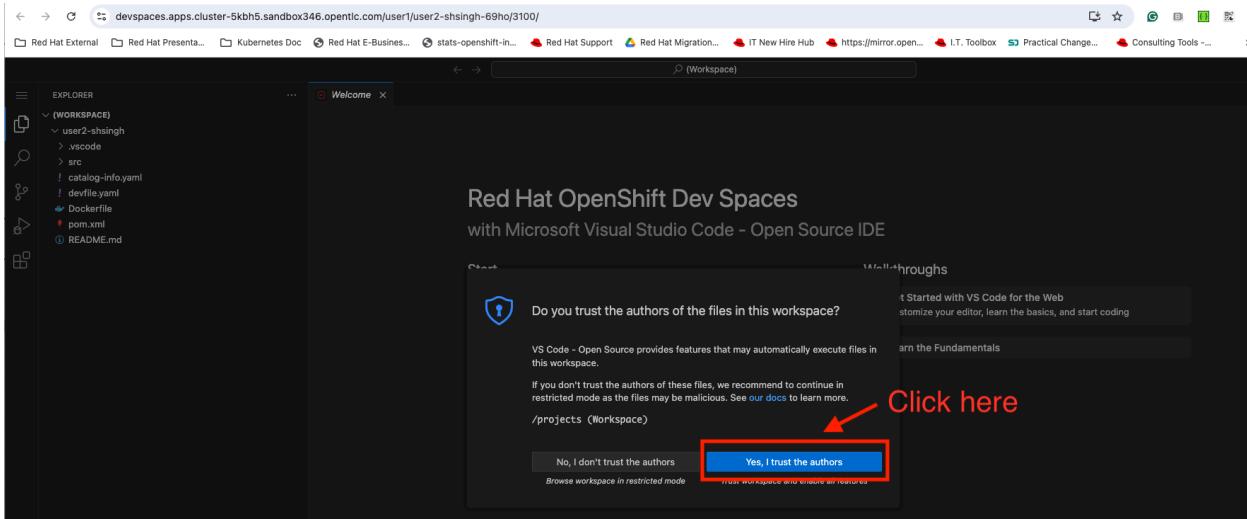
ArgoCD App	Namespace	Instance	Server	Revision	Last deployed	Sync status	Health status
user2-shsingh	user2-shsingh	main	https://kubernetes.default.svc	d2aada7	8/08/2024, 13:52:25	Synced	Progressing
user2-shsingh-build	user2-shsingh	main	https://kubernetes.default.svc	d2aada7	8/08/2024, 13:52:30	Synced	Healthy

Verify

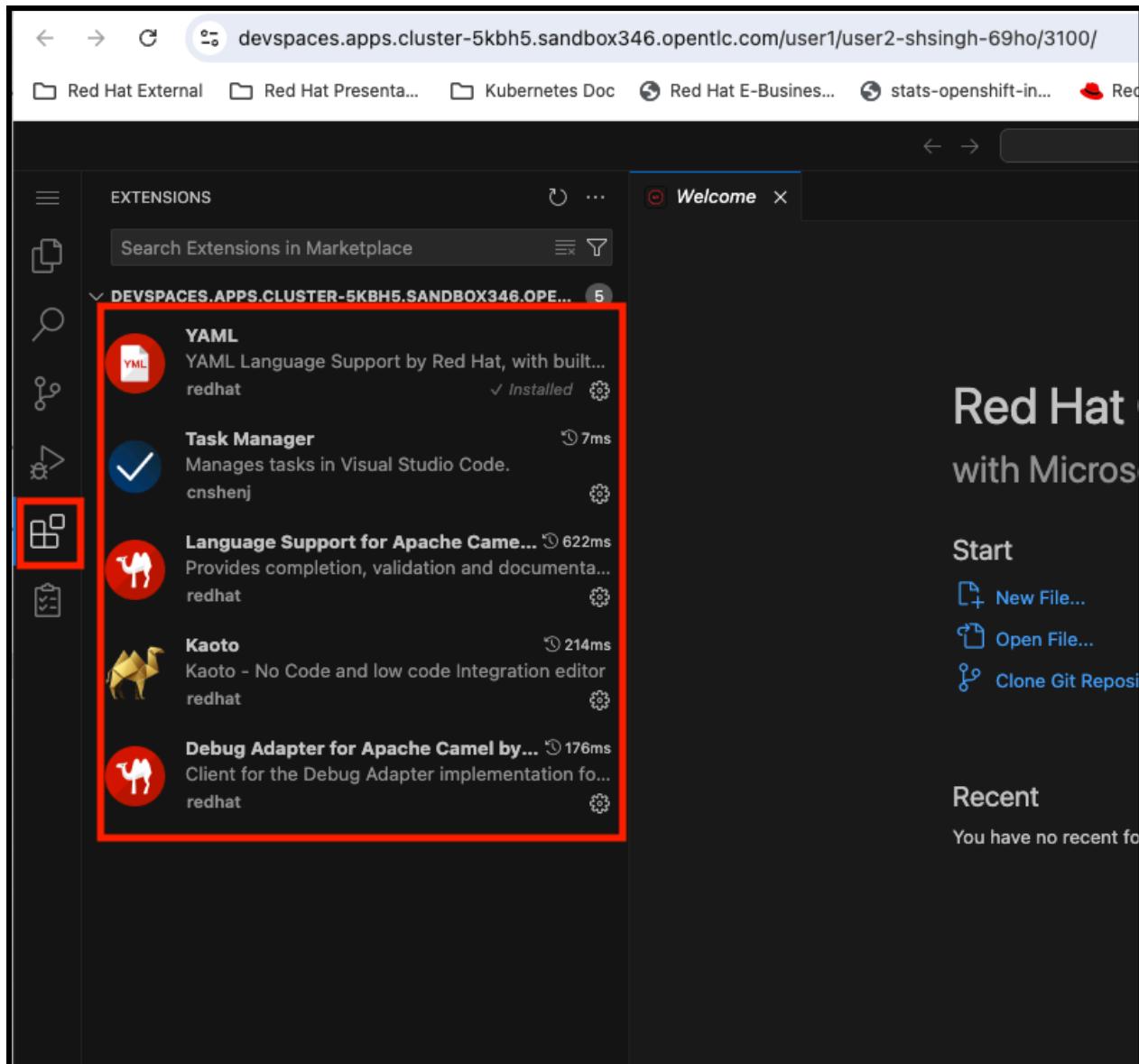
4 Open the new workspace from the developer Hub.



4.1 After waiting a few minutes for OpenShift Dev Spaces to finish setting up your workspace, You click the button Yes, I trust the authors.



4.2. Verify all the extensions have been installed correctly.



5 Module 1

5.1 Update the application.properties

```

application.properties 2 x
user2-shsingh > src > main > resources > application.properties
1 # the name of Camel
2 camel.springboot.name = MyCamel
3
4 camel.component.servlet.mapping.context-path=/camel/*
5 camel.component.servlet.name=CamelServlet
6
7 # what to say
8 greeting = Hello World
9
10
11 camel.component.kafka.security-protocol = SASL_SSL
12 camel.component.kafka.sasl-mechanism= SCRAM-SHA-256
13 camel.component.kafka.ssl-truststore-location = /opt/secrets/clienttruststore.jks
14 camel.component.azure-storage-blob.credential-type= SHARED_ACCOUNT_KEY
15
16 #Replace this
17 camel.component.kafka.brokers=
18 camel.component.kafka.sasl-jas-config=
19 camel.component.kafka.ssl-truststore-password =
20 camel.component.azure-storage-blob.access-key=
21 #####
22
23 camel.springboot.routes-include-pattern = classpath:route/Module1.camel.yaml
24
25
26
27 topicName.json=user2-shsingh-json
28
29 consumergroup=user2-shsingh-consumer
30
31 blobName=user2-shsingh-xml
32
33
34 # how often to trigger the timer
35 timer.period = 2000
36
37 # to automatic shutdown the JVM after a period of time
38 #camel.springboot.duration-max-seconds=60
39 #camel.springboot.duration-max-messages=100
40

```

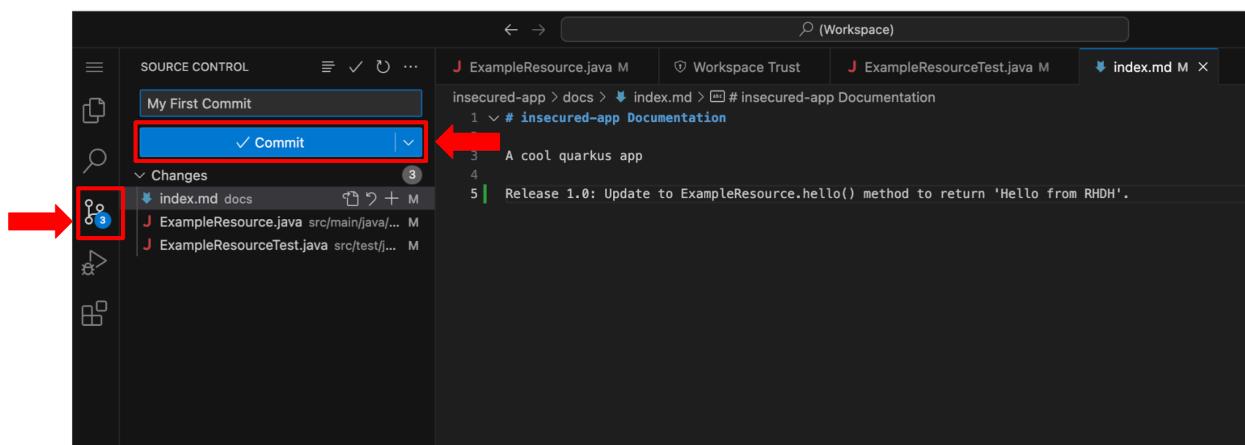
~~~~

camel.component.kafka.brokers=my-cluster-kafka-route1-bootstrap-amq-stre  
 ams-kafka.apps.shailendra14k.in:443  
 camel.component.kafka.sasl-jas-config=  
 org.apache.kafka.common.security.scram.ScramLoginModule required  
 username="console-cluster-user1"  
 password="JkP7hGT7rpJd1DdBQ3zb1DTUZ6Gau0zH";  
 camel.component.kafka.ssl-truststore-password = indigo  
 camel.component.azure-storage-blob.access-key=  
 Y/fgFLVIPUGJ6MfQlWLxPWaILEE+BXV2FtzQwKwePp2XHKpF2djTaaNfc7utIHxD+m3OAxs8  
 Ih4i+ASTY7QFJg==  
~~~~~

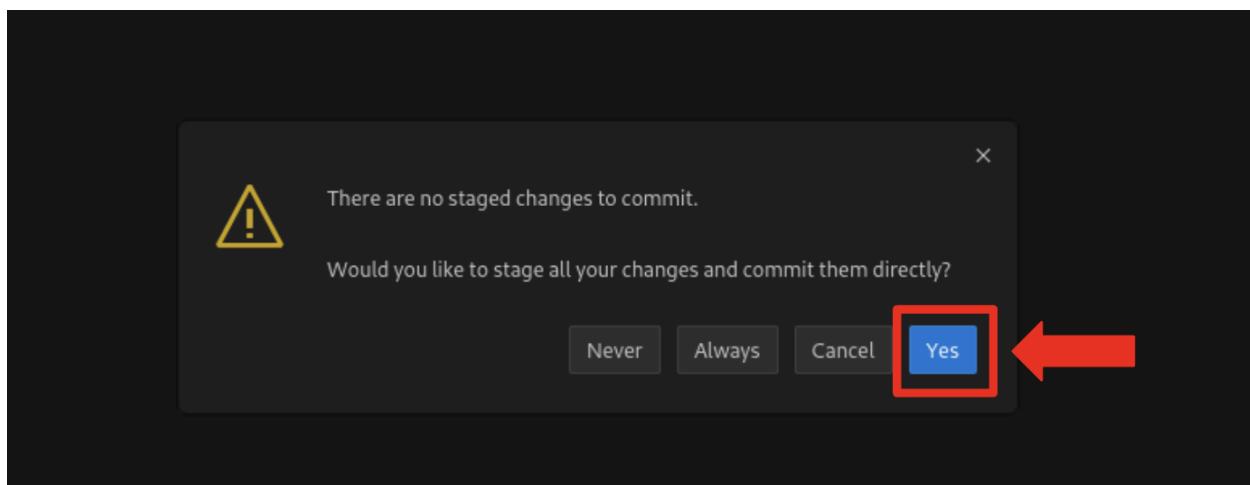
5.2 Commit the Changes

5.2.1. You click on the Source Control icon located in the left menu.

5.2.2. Then, you enter the commit message "updated application properties" and click on the Commit button to finalize your changes.

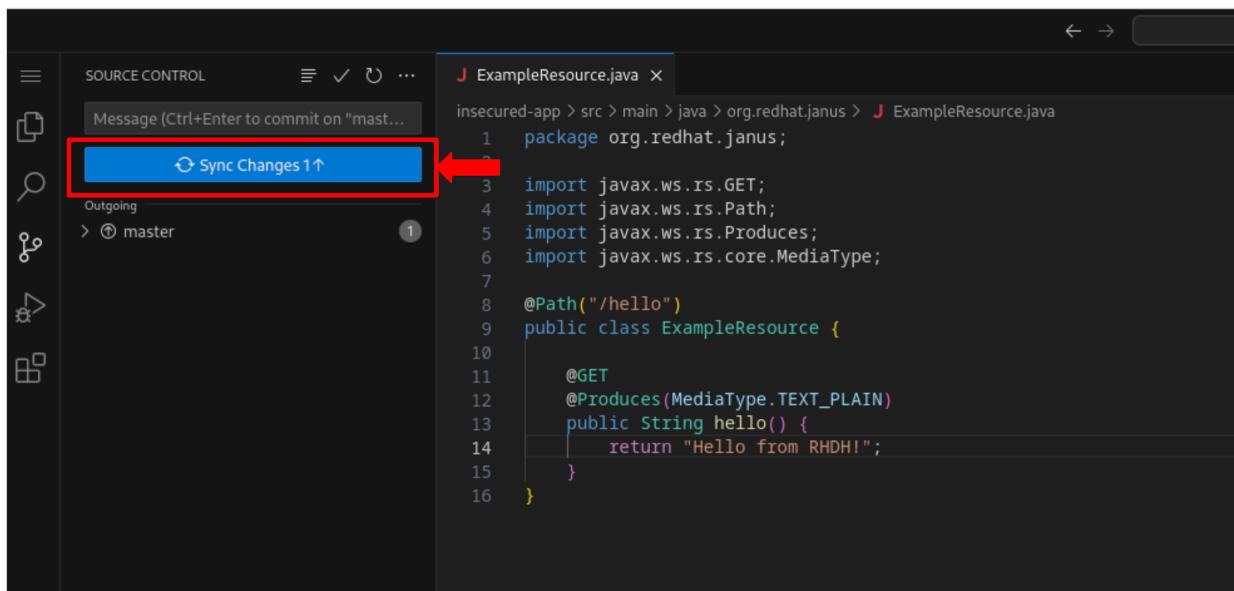


5.2.3. In the pop-up window that follows, you click Yes to stage your changes.

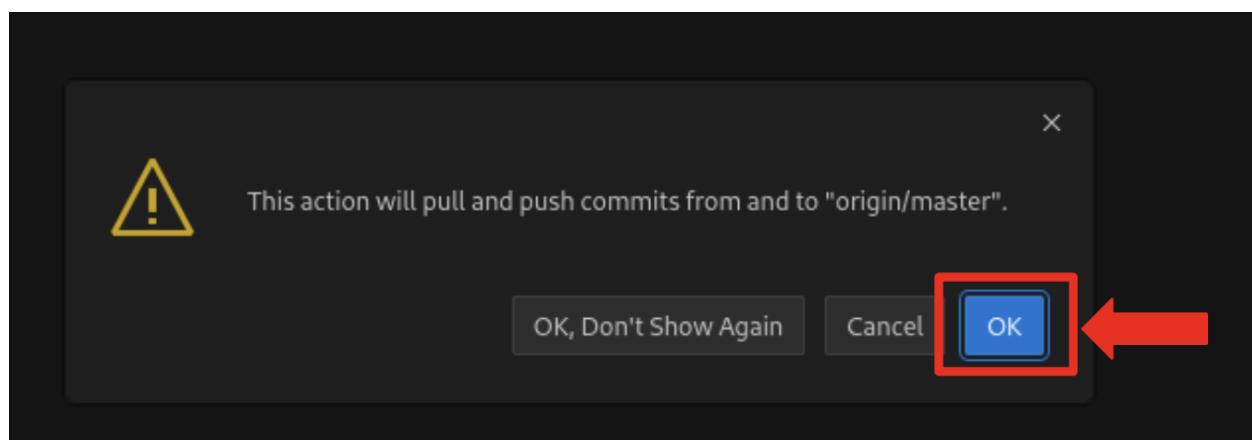




5.2.4. Finally, you click on the Sync Changes button.



6.2.5. In the pop-up that follows, you click OK to push your changes and complete the process.



In Case the commit fails like the below

The screenshot shows a code editor interface with a yellow border. In the center, there is a modal dialog box with a red close button. The dialog contains the following text:

```

Failed to authenticate to git remote:
https://gitlab-gitlab.apps.cluster-5kbh5.sandbox346.opentlc.com/development/user2-shsingh.git/

```

Below the modal are three buttons: "Show Command Output", "Cancel", and "Open Git Log".

Delete all the workspaces from the Devspace URL and start the new workspace from the same software catalogue.

5.3 Verify the pipeline

The screenshot shows the Red Hat Developer Hub interface. On the left, there is a sidebar with various navigation options like Home, Catalog, APIs, Learning Paths, Create..., Tech Radar, Docs, and Clusters. The main area displays a component named "user2-shsingh" with a star icon. Below the component name, there are tabs for Overview, Topology, Issues, Pull/Merge Requests, CI (which is selected), CD, Kubernetes, API, Dependencies, and Docs. Under the CI tab, there is a section titled "Pipeline Runs". It shows a single pipeline run with the name "PLR user2-shsingh-run-v2knq". The status of the run is "Succeeded" with a green bar. The duration of the run was "4 minutes 57 seconds". Below the run, there is a diagram showing the pipeline steps: "git-clone" (1/1), "maven" (1/1), and "s2i-java" (1/1). At the bottom of the pipeline run section, there are several small icons for search, refresh, and other actions.

5.4 Open the application

URL:-

`https://<PROJECT-NAME>-<PROJECT-NAME>.<CLUSTER-DOMAINNAME>/camel/demo`

SAMPLE URL :-

`https://user2--shsingh03.apps.cluster-5kbh5.sandbox346.opentlc.com/camel/demo`



6 Module 2

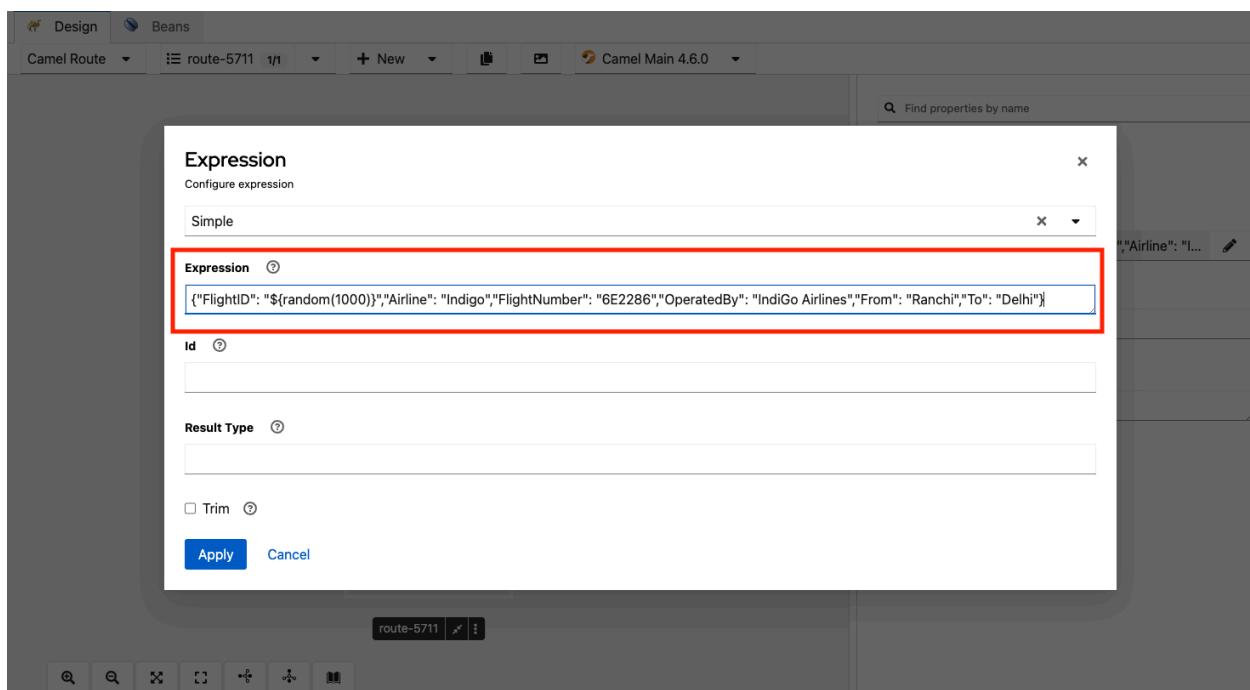
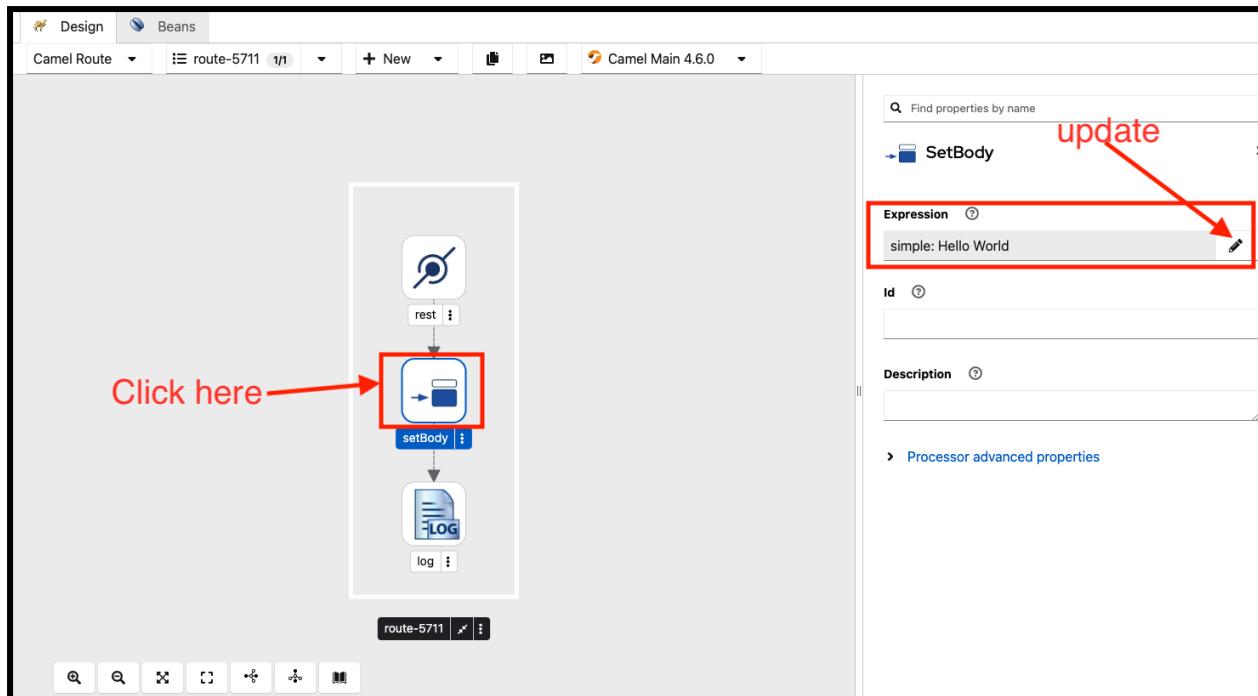
6.1 Update Expression of SetBody

6.1.1. Go to `src/main/java/resource/Module1.camel.yaml`

6.2.2. Update the `setBody` component with below expression

Expression:-

```
{"FlightID": "${random(1000)}", "Airline": "Indigo", "FlightNumber":  
"6E2286", "OperatedBy": "IndiGo Airlines", "From": "Ranchi", "To": "Delhi"}
```



6.2 Commit the Changes

[Follow Module1 steps](#)

6.3 Verify the pipeline has been completed successfully.

[Follow Module1 steps](#)

6.4 Verify the Output

URL :-

<https://<PROJECT-NAME>-<PROJECT-NAME>.<CLUSTER-DOMAINNAME>/camel/demo>

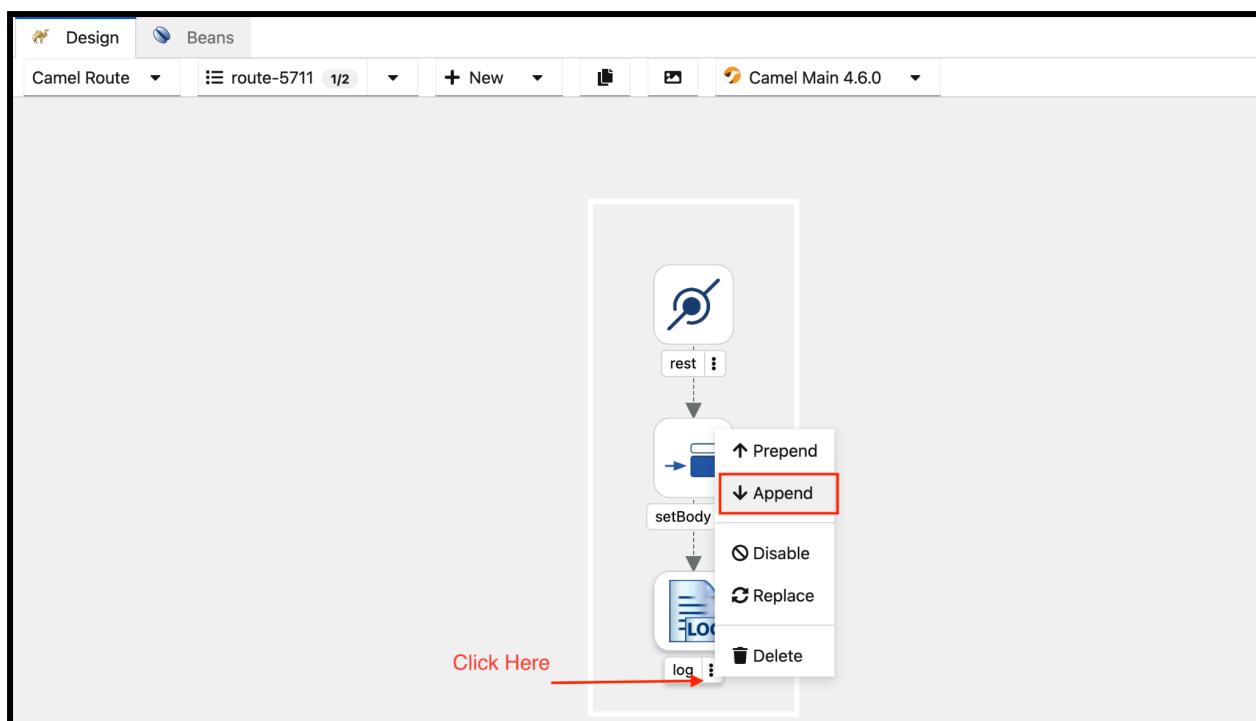
SAMPLE URL :-

<https://shsingh03-shsingh03.apps.cluster-5kbh5.sandbox346.opentlc.com/camel/demo>

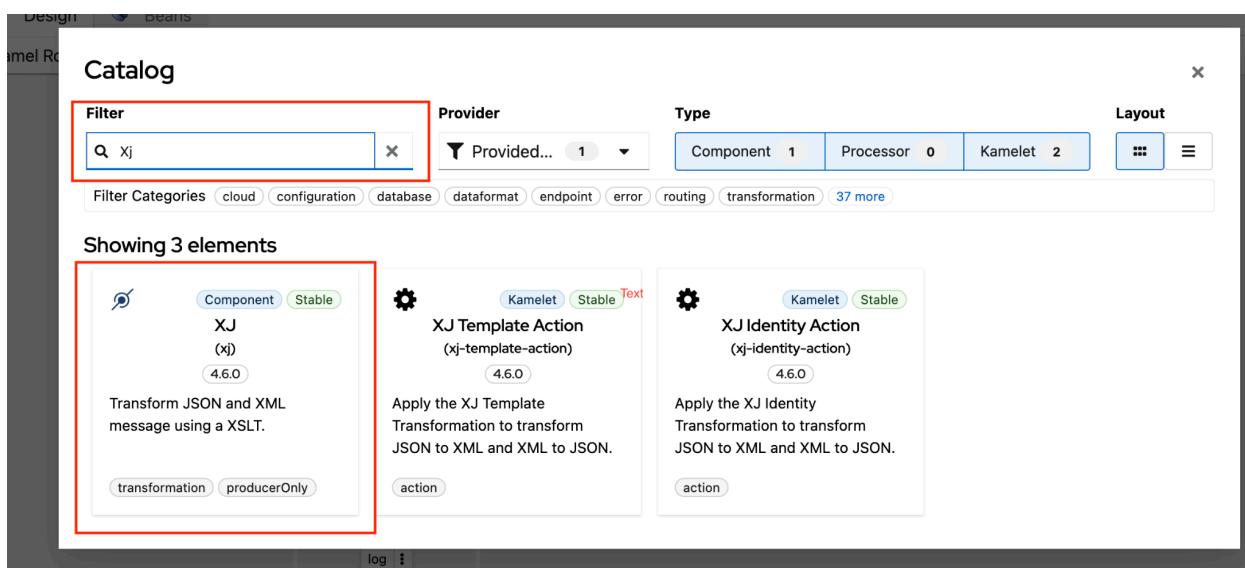


7. Module 3

7.1. Click on the 3 dot of the log component and select append

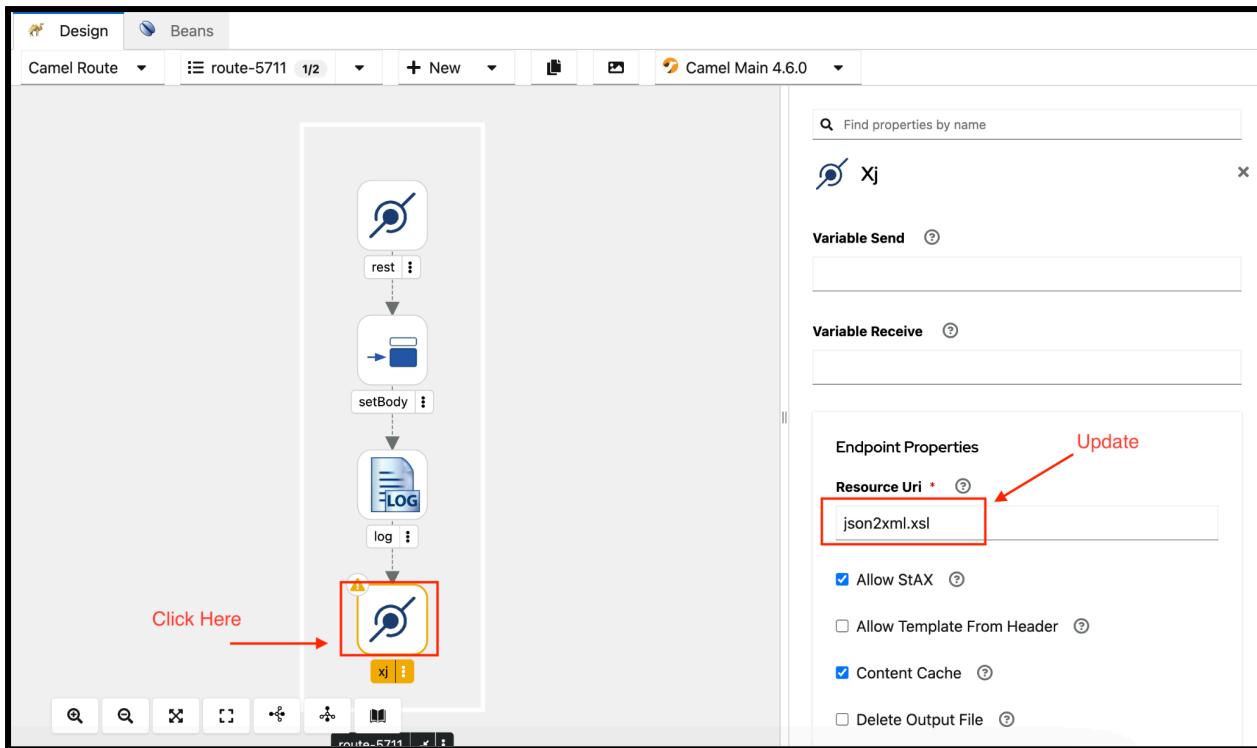


7.2. Select the xj component from the catalog

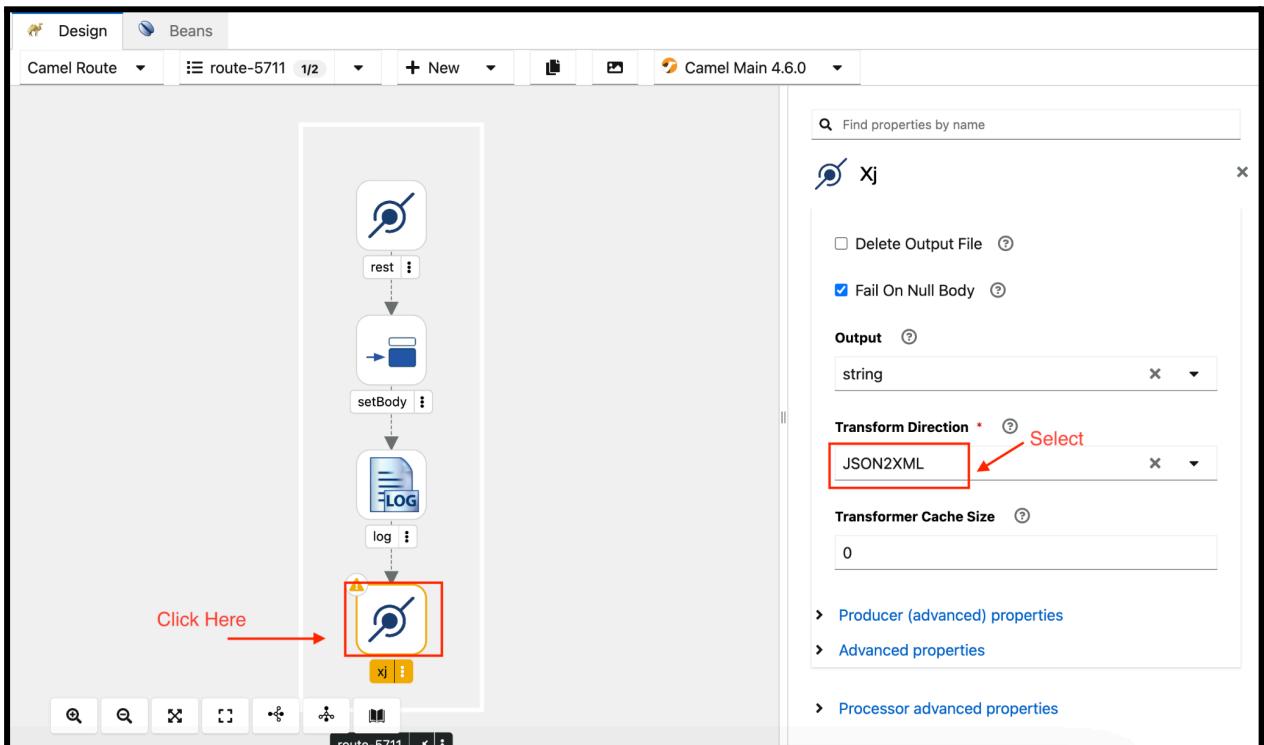


7.2.1 Update the below configuration for the xj component

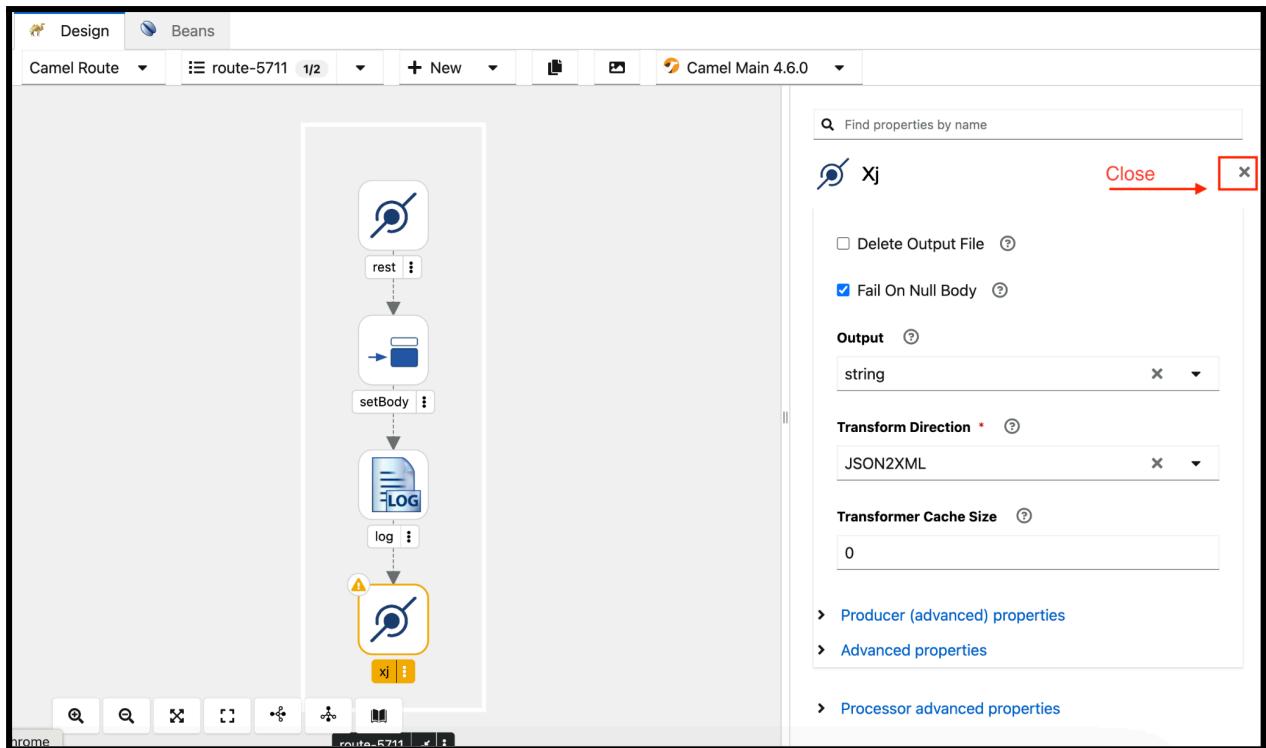
- Resource Uri --> json2xml.xsl



- Transform Direction ---> JSON2XML



- Close



7.3 Commit the Changes

[Follow Module1 steps](#)

7.4 Verify the pipeline has been completed successfully.

[Follow Module1 steps](#)

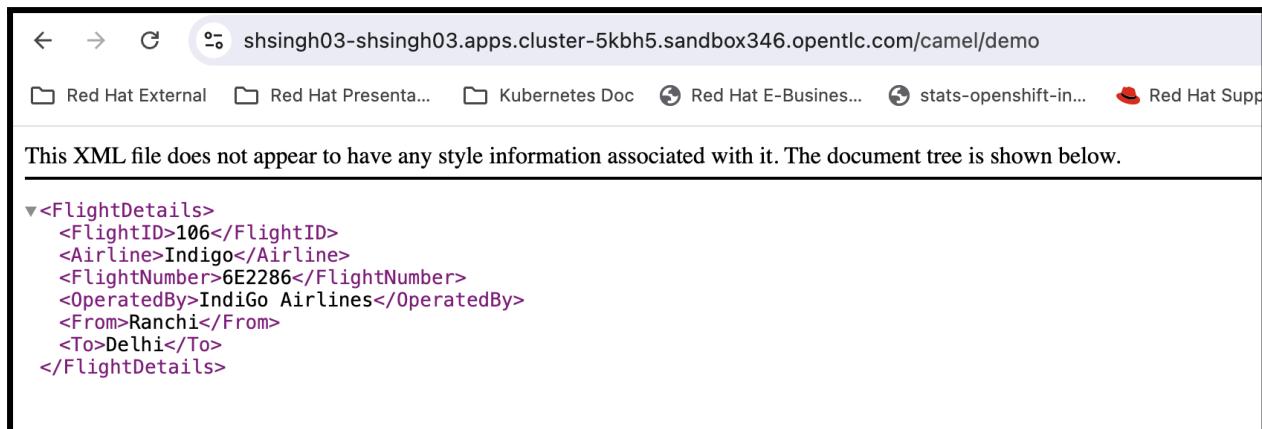
7.5 Verify the Output

URL:-

<https://<PROJECT-NAME>-<PROJECT-NAME>.<CLUSTER-DOMAINNAME>/camel/demo>

SAMPLE URL :-

<https://shsingh03-shsingh03.apps.cluster-5kbh5.sandbox346.opentlc.com/camel/demo>

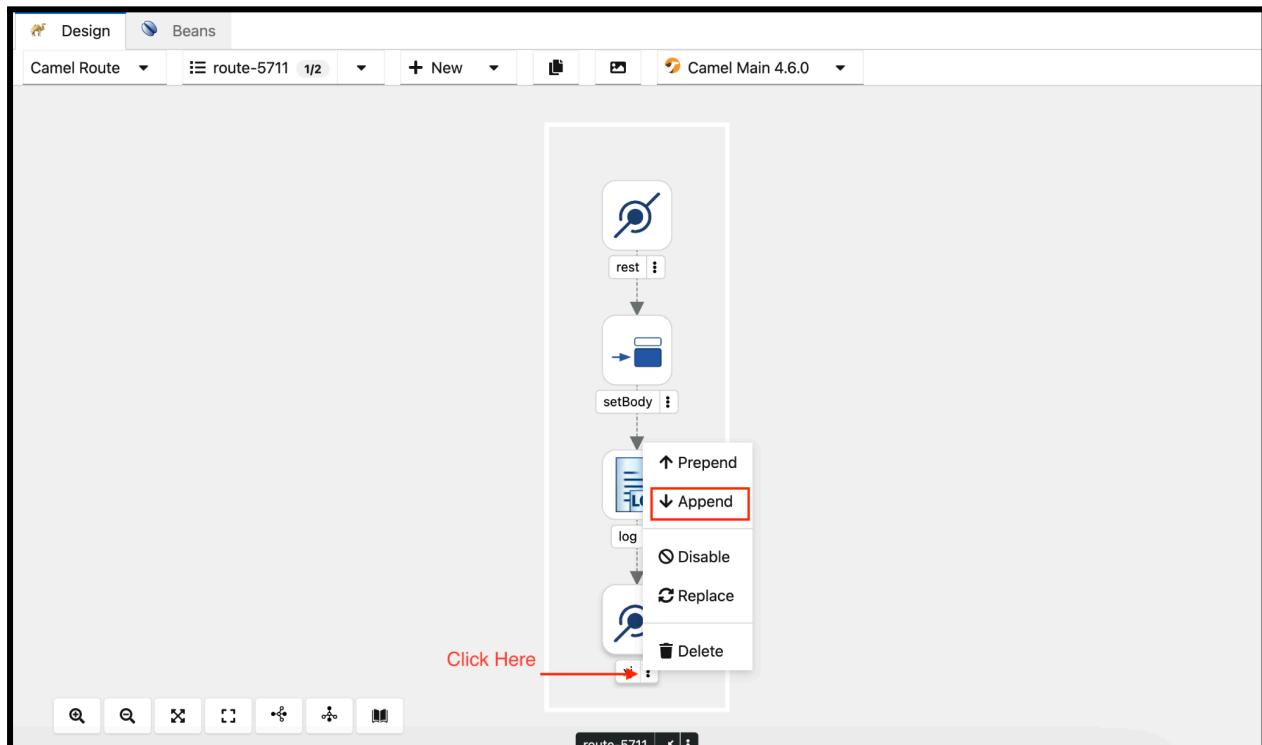


This XML file does not appear to have any style information associated with it. The document tree is shown below.

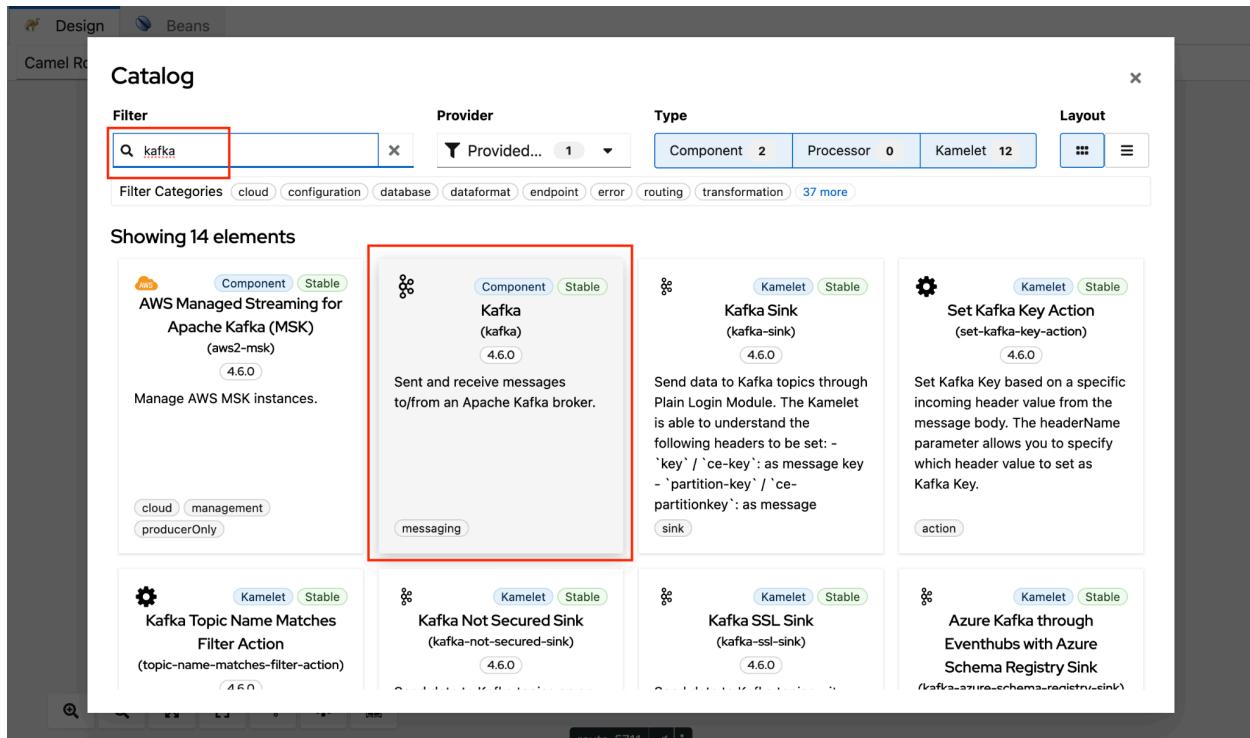
```
<FlightDetails>
  <FlightID>106</FlightID>
  <Airline>Indigo</Airline>
  <FlightNumber>6E2286</FlightNumber>
  <OperatedBy>IndiGo Airlines</OperatedBy>
  <From>Ranchi</From>
  <To>Delhi</To>
</FlightDetails>
```

8 Module 4

8.1. Click on the 3 dot of the XJ component and select append

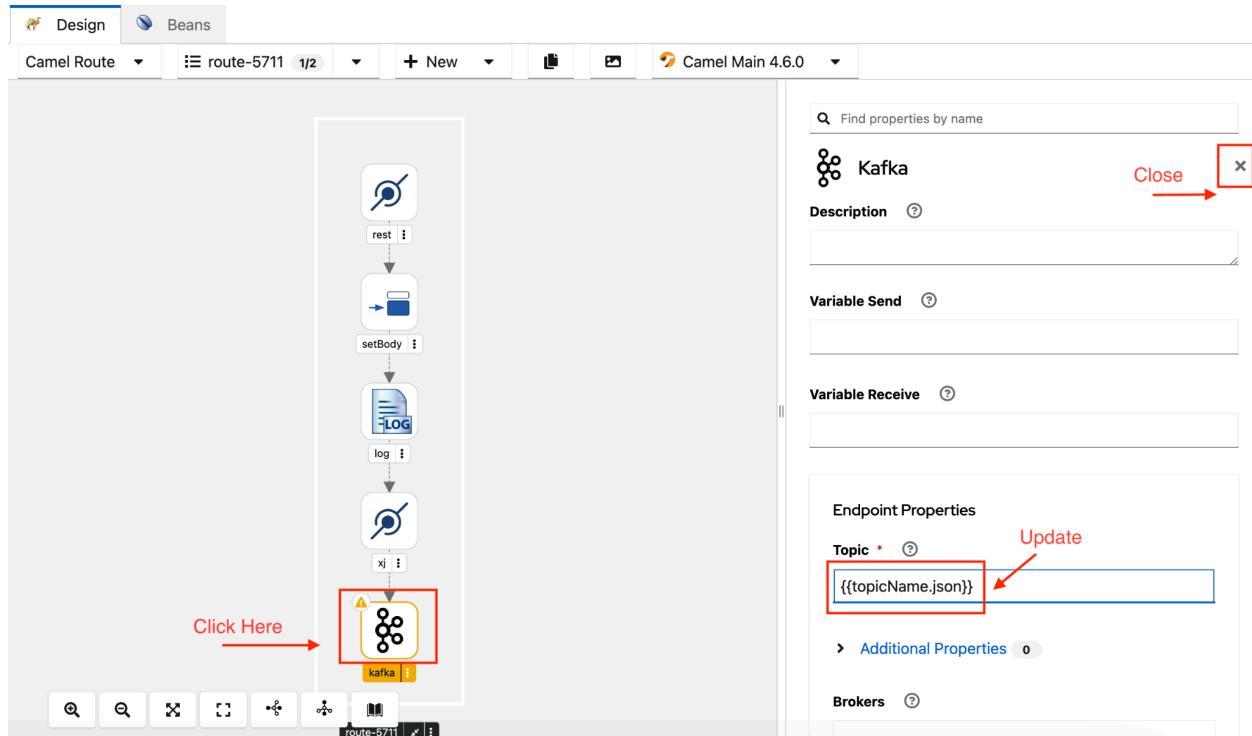


8.2.Select the kafka component from the catalog



8.2.1 Update properties for the Kafka component and close

- Topic --> {{topicName.json}}



8.3 Commit the Changes

[Follow Module1 steps](#)

8.4 Verify the pipeline has been completed successfully.

[Follow Module1 steps](#)

8.5 Verify the Output

AMQ Stream (Kafka) access will be with the Instructor.

Topic with the <projectName>-json will be created and will have all the messages. The screenshot below is for user2.

This is a Tech Preview version. You are logged in as an anonymous user with read-only access.

Kafka clusters > my-cluster > Topics > user1-sumit-json

user1-sumit-json

Messages Partitions Consumer groups Configuration

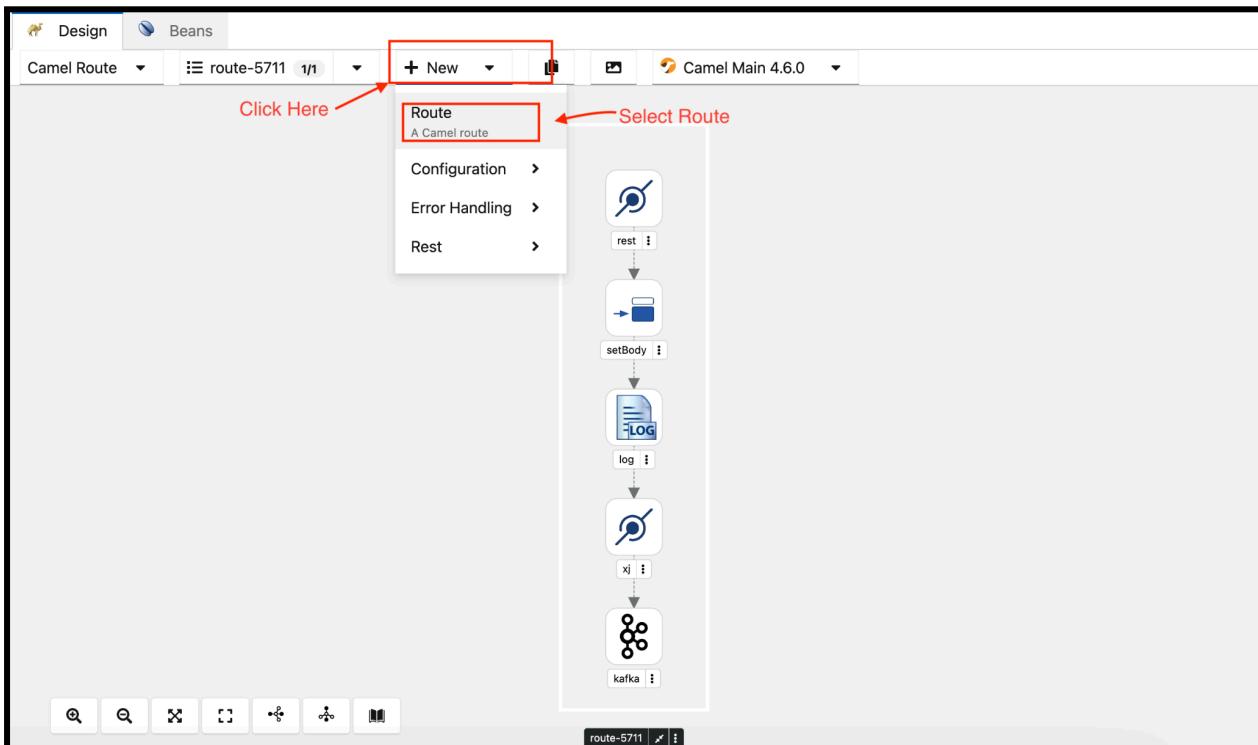
messages=latest retrieve=50

Timestamp (UTC)	Offset	Key	Value
8/8/24, 11:28:17 AM	0	No data	<?xml version="1.0" encoding="UTF-8"?> <FlightDetails> <FlightID>620</FlightID> <Airline>Indigo</Airline> <FlightNumber>6E2286</FlightNumber> <OperatedBy>IndiGo A

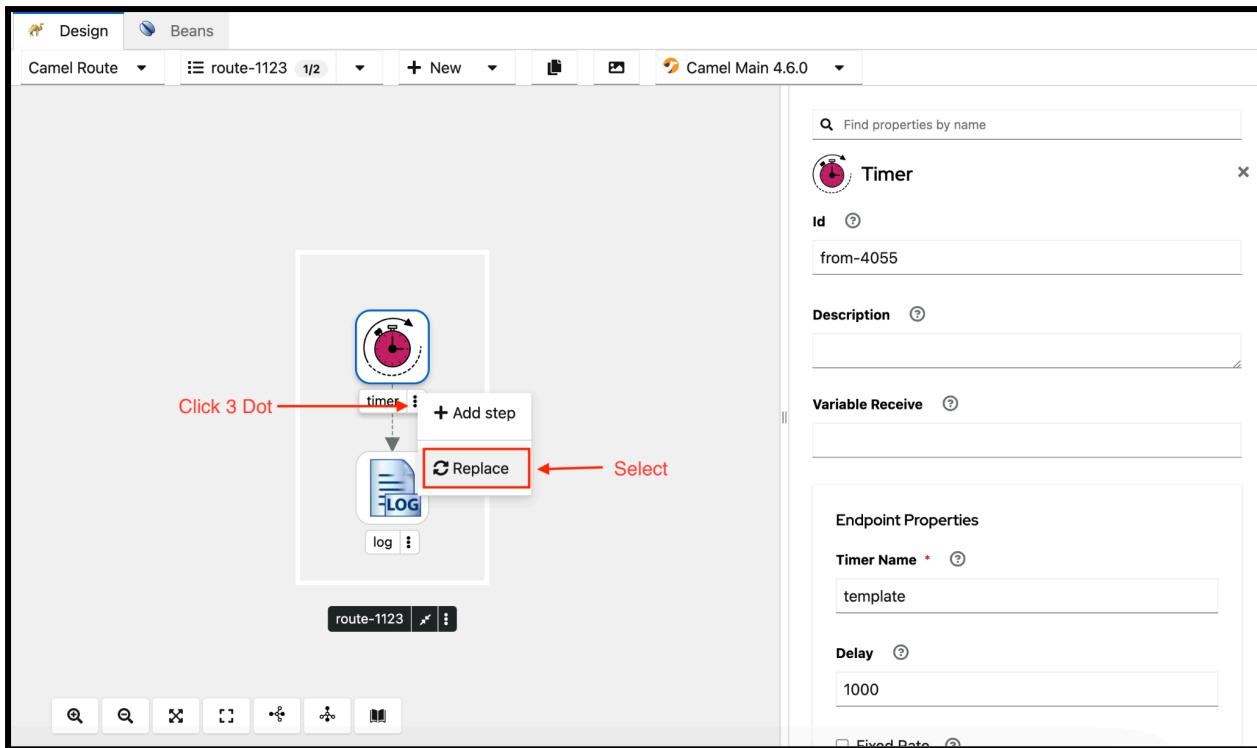
Partition 0

9 Module 5

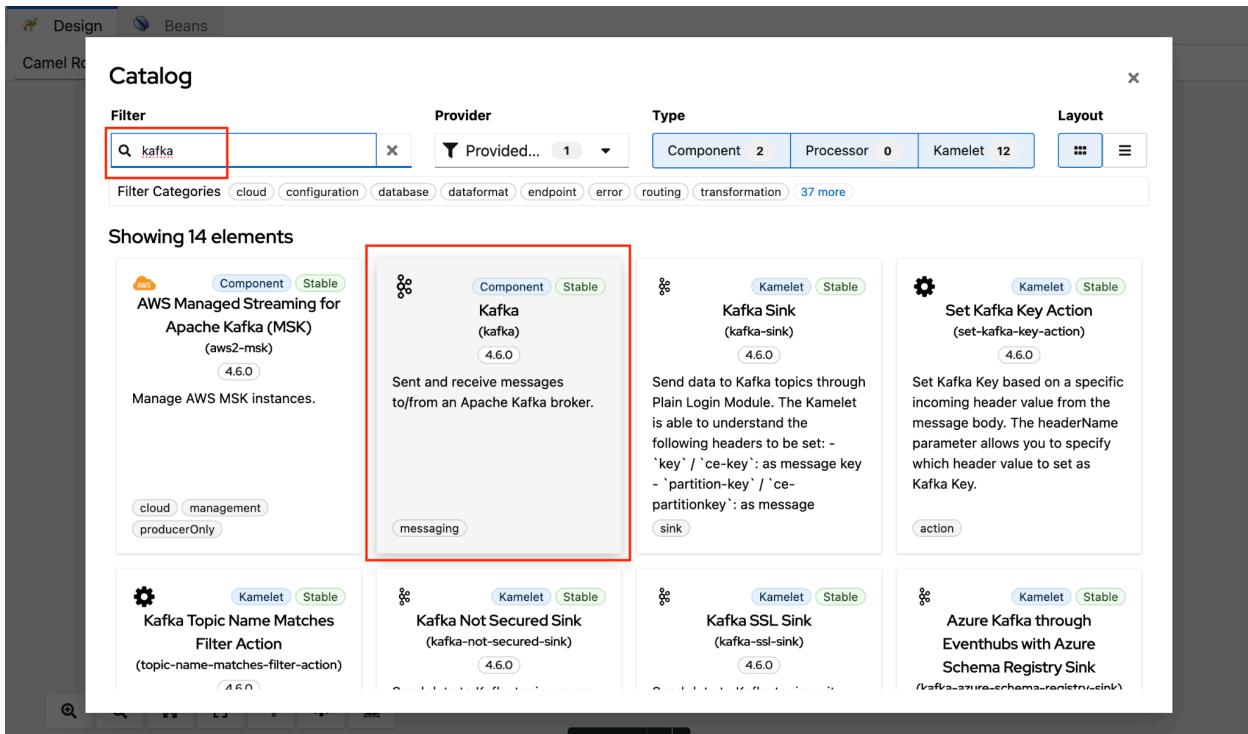
9.1. Create a new route



9.2. Replace Timer with Kafka component.

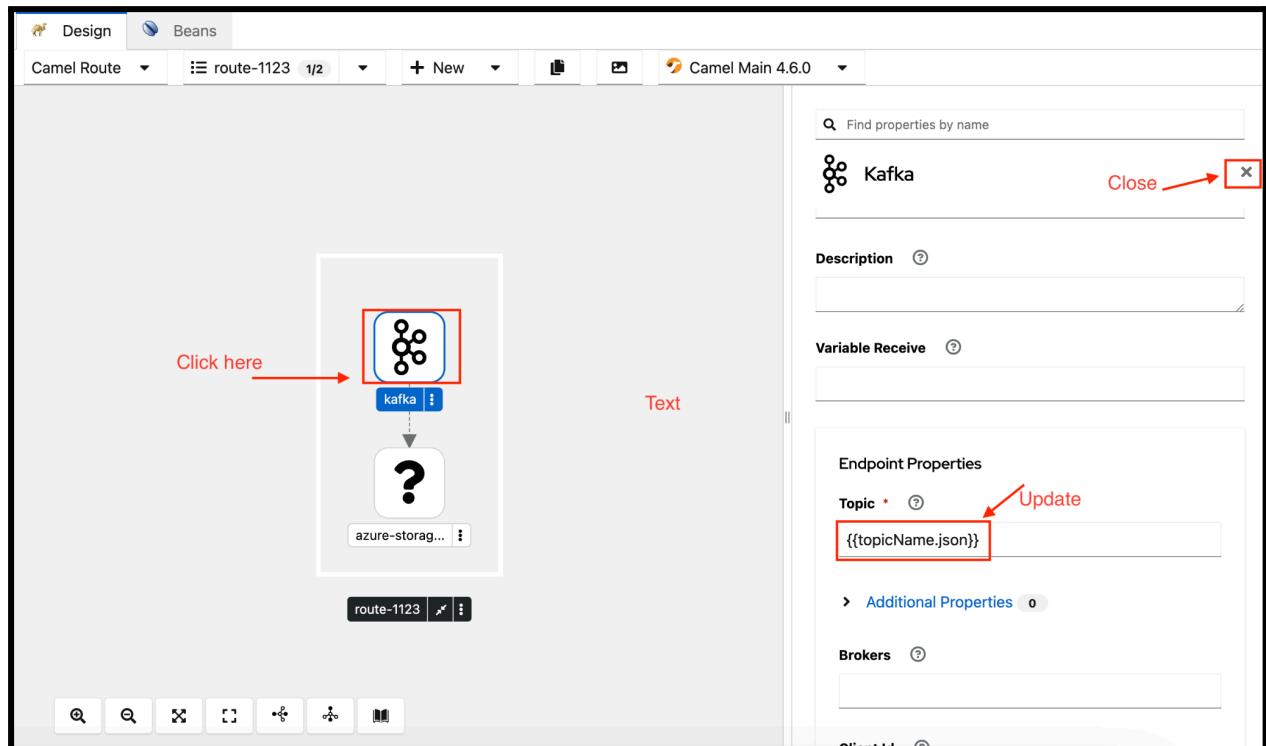


9.3.Select the kafka component from the catalog

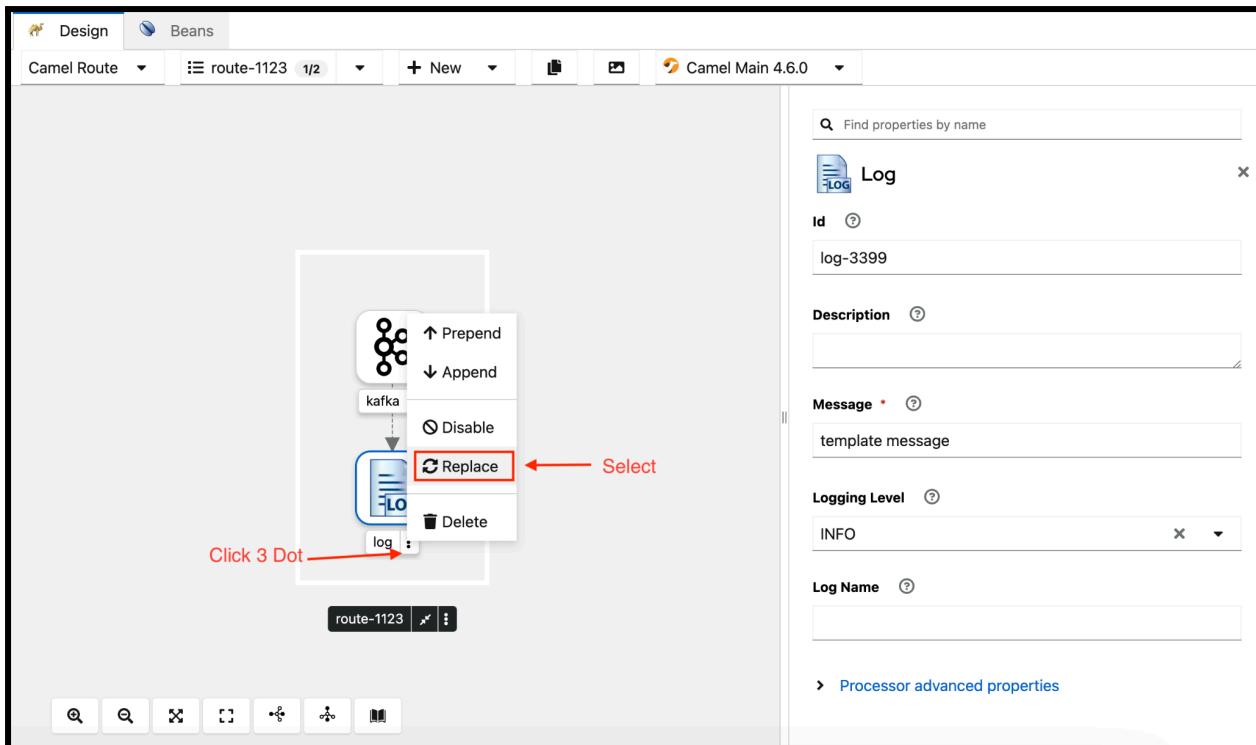


9.3.1 Update properties for the Kafka component and close

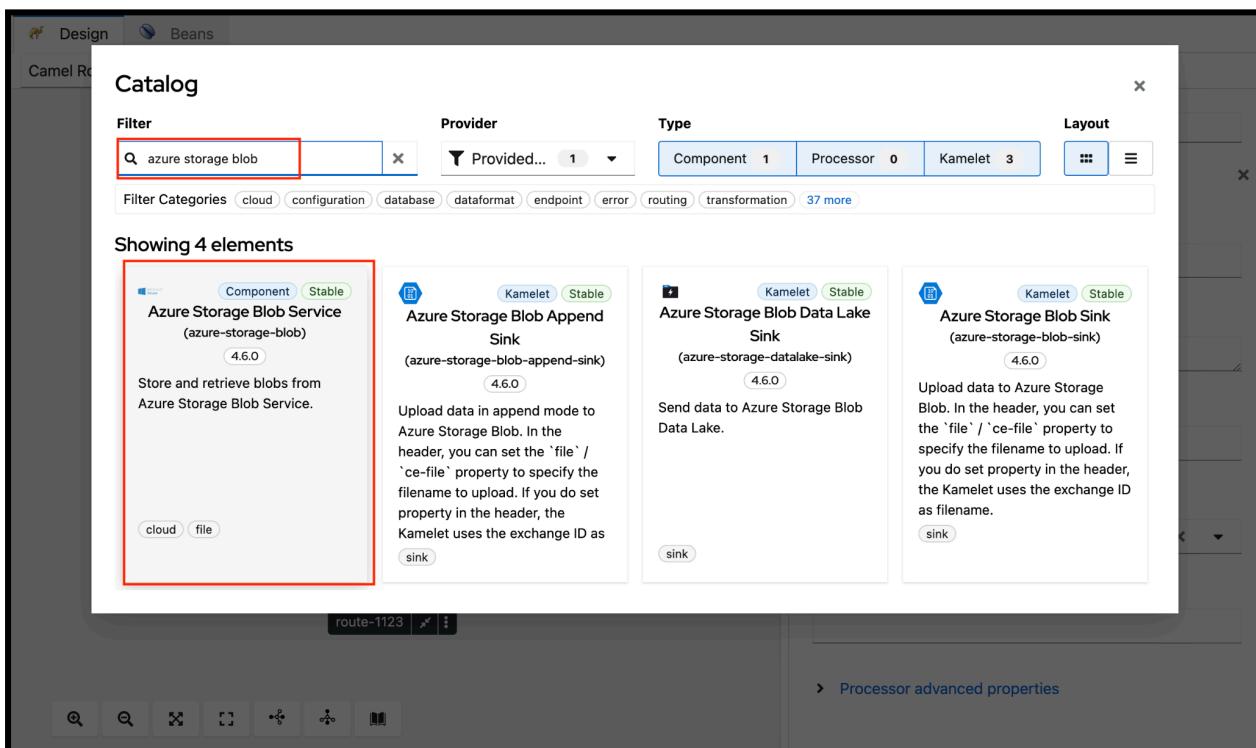
- Topic --> {{topicName.json}}



9.4. Replace Log with AzureBlob Storage component.

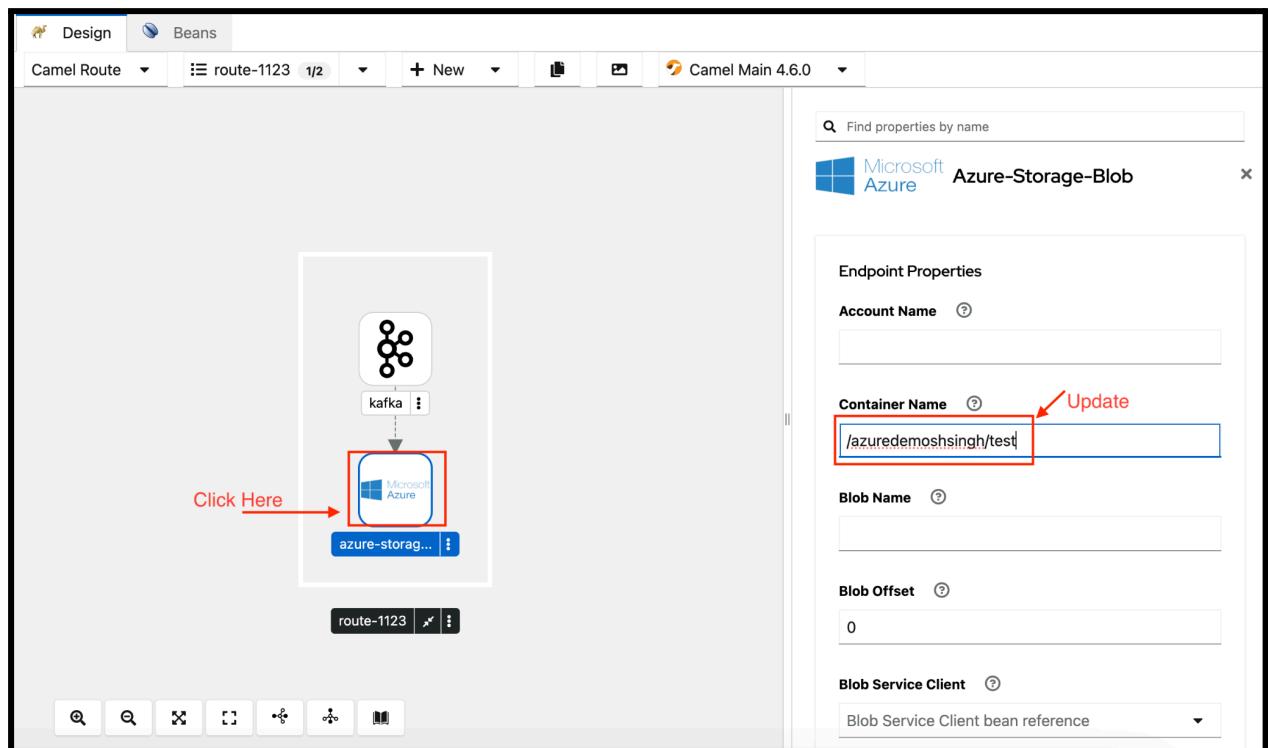


9.5. Select the AzureBlob Storage component from the catalog

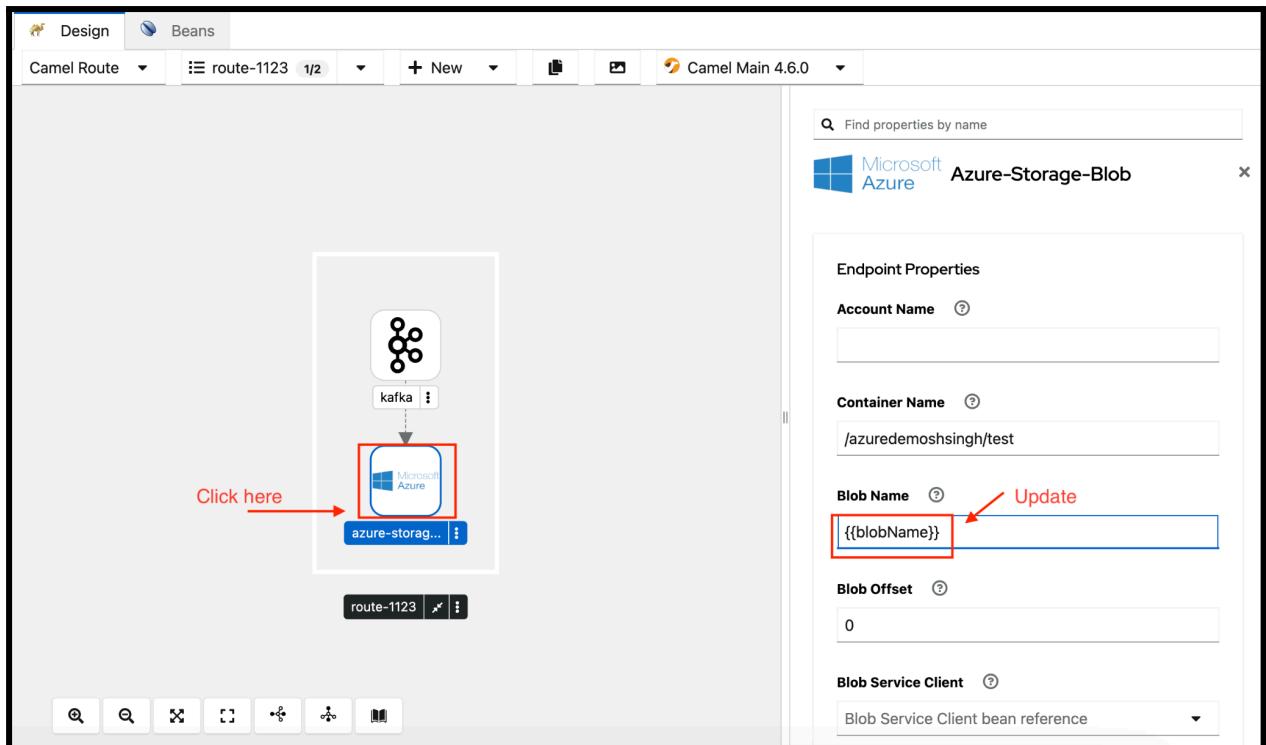


9.5.1 Update parameters for the azure-storage-blob component

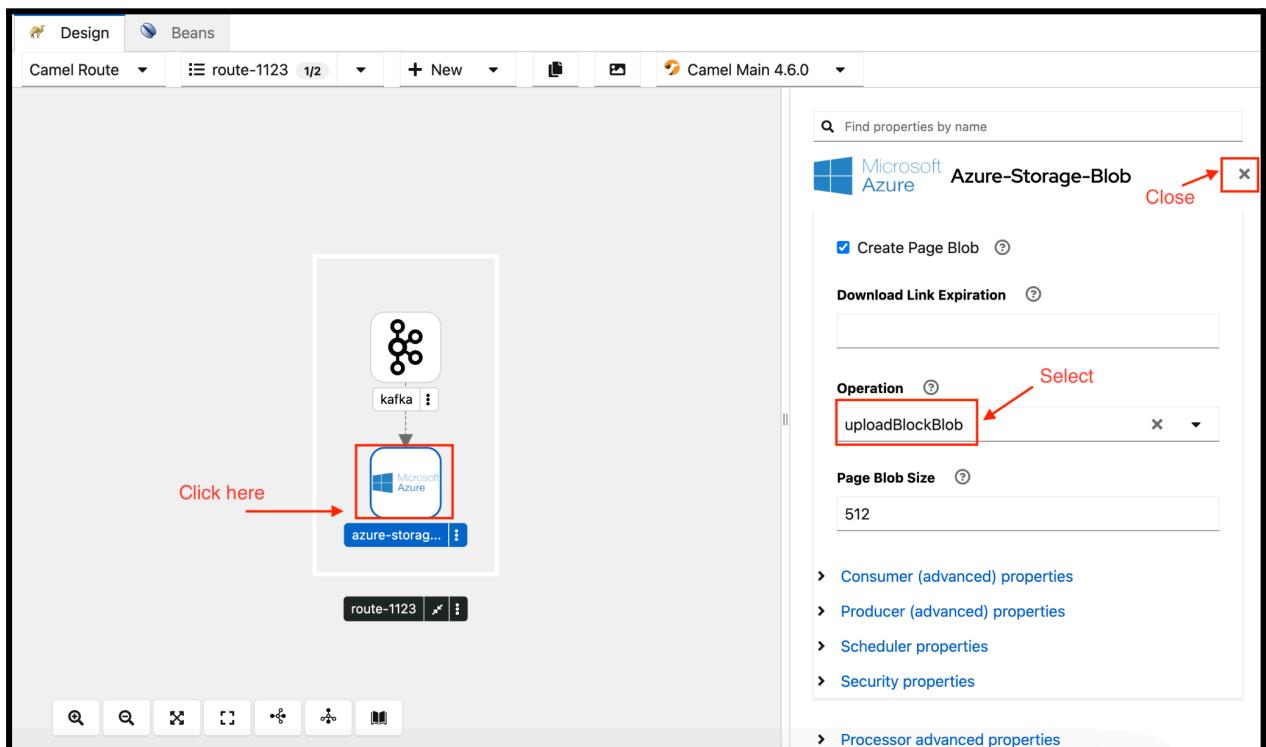
- Container Name ---> /azuredemoshsingh/test



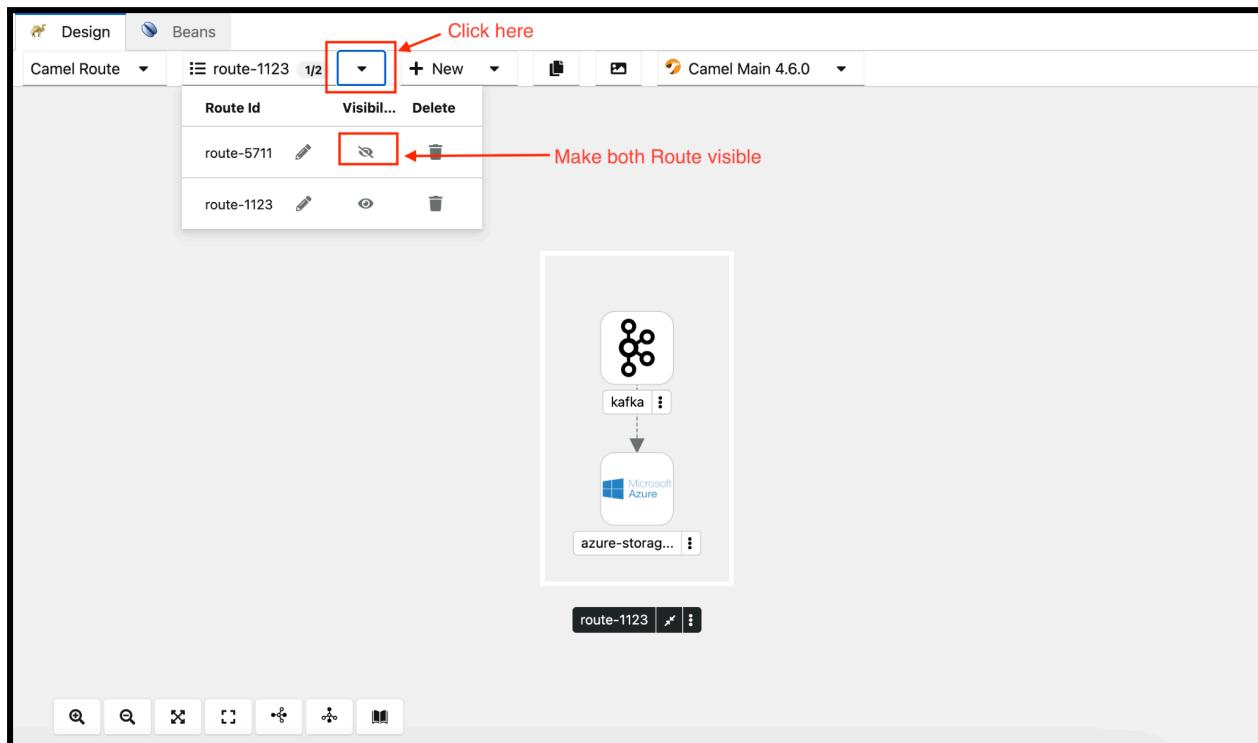
- Blob Name ---> {{blobName}}



- Operation ---> uploadBlockBlob



9.6. Select both the route



9.7 Commit the Changes

[Follow Module1 steps](#)

9.8 Verify the pipeline has been completed successfully.

[Follow Module1 steps](#)

9.9 Verify the Output

Azure Storage account access will be with the Instructor.

File with the name <projectName>-xml will be created on the azure storage. The screenshot below is for user2.

10. Update the DevSpace plugin

Each of the following sections provides a brief description and a list of recommended actions. Additional explanations are provided where additional clarity is required.

10.1. Login to the openshift console using admin credentials.

UserName:- admin

Password:- <SameAsUser>

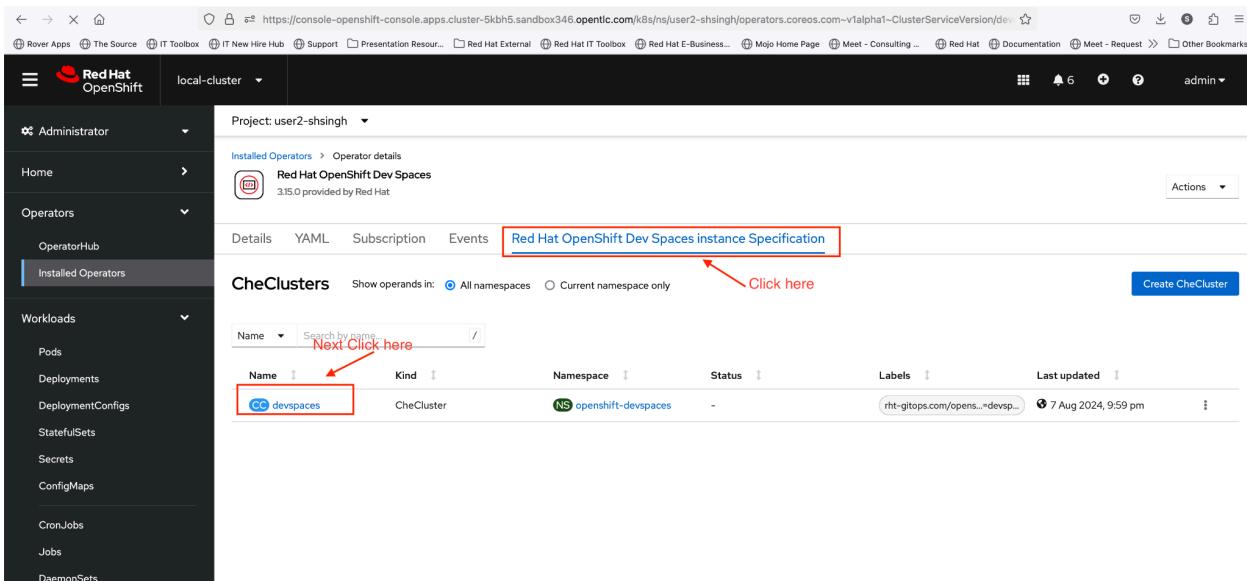
URL:- <https://console-openshift-console.<CLUSTER-DOMAINNAME>>

SAMPLE URL :- <https://console-openshift-console.apps.cluster-5kbh5.sandbox346.opentlc.com>

10.2. Go to the Installed OperatorPage and look for Red Hat OpenShift Dev Spaces

Name	Managed Namespaces	Status	Last updated	Provided APIs
Red Hat OpenShift Dev Spaces	All Namespaces	Succeeded Up to date	7 Aug 2024, 9:59 pm	Red Hat OpenShift Dev Spaces instance Specification
DevWorkspace Operator	All Namespaces	Succeeded Up to date	7 Aug 2024, 9:59 pm	DevWorkspace DevWorkspaceTemplate DevWorkspaceOperatorConfig
GitLab Runner	All Namespaces	Succeeded Up to date	7 Aug 2024, 9:58 pm	GitLab Runner
Red Hat OpenShift GitOps	All Namespaces	Succeeded Up to date Plugin available	7 Aug 2024, 9:27 pm	Argo CD AnalysisRun AnalysisTemplate Application View 6 more...

10.3 Update the devspace cheCluster Yaml configurations.



Name	Kind	Namespace	Status	Labels	Last updated
devspaces	CheCluster	openshift-devspaces	-	rht-gitops.com/opens...=devsp...	7 Aug 2024, 9:59 pm

10.4. Update the pluginRegistry configurations.

```
oc patch checluster devspaces --type=merge -p '{"spec": {"components": {"pluginRegistry": {"openVSXURL": "https://open-vsx.org"} }}}' -n openshift-devspaces
```

pluginRegistry:

openVSXURL: '<https://open-vsx.org>'

The screenshot shows the Red Hat OpenShift console interface. On the left, there's a sidebar with navigation links like Home, Operators, Workloads, and Cron.Jobs. The main area displays a YAML configuration for the 'devspaces' operator. A specific line of code, 'openVSXURL: "https://open-vsx.org"', is highlighted with a red box and an annotation pointing to it with the text 'Update this registry'. At the bottom of the YAML editor, there are three buttons: 'Save' (highlighted with a red box), 'Reload', and 'Cancel'. To the right of the YAML editor, there's a large preview pane showing the current state of the cluster.

```

129   labels:
130     rht-gitops.com/openshift-gitops: devspaces
131   spec:
132     components:
133       cheCluster:
134         debug: false
135         logLevel: INFO
136         dashboard:
137           logLevel: ERROR
138           devWorkspace: {}
139           devfileRegistry: {}
140         imagePuller:
141           enable: false
142           spec: {}
143           metrics:
144             enable: true
145           pluginRegistry:
146             openVSXURL: "https://open-vsx.org"
147           environments: {}
148           startTimeoutSeconds: 300
149           security: {}
150           secondsOfRunBeforeTiling: -1
151           maxNumberOfWorkspacesPerUser: -1
152           containerBuildConfiguration:
153             openShiftSecurityContextConstraint: container-build
154             defaultEditor: che-incubator/che-code/latest
155             maxNumberOfRunningWorkspacesPerUser: 2
156

```

10.5. Save the configuration and verify the changes after reload.