ACEv11.0.0.2 On

IBM Cloud Private 3.1

Devops driven integration to micro services principles

V1.1 Draft

Dave Arnold

Do Nguyen

Peter Jessup

February 11, 2019

Table of Contents

[Introduction 5](#_Toc813003)

[Scenario Overview Diagram 5](#_Toc813004)

[Overview Description 5](#_Toc813005)

[The Scenario 5](#_Toc813006)

[The Integration Micro Services 5](#_Toc813007)

[The docker images 6](#_Toc813008)

[The Personas 6](#_Toc813009)

[Customized Jenkins Pipelines 6](#_Toc813010)

[Reference Section 7](#_Toc813011)

[Summary of Github Repositories 7](#_Toc813012)

[Summary of Dockerhub Images 7](#_Toc813013)

[ACE Integration Liveliness Probe 8](#_Toc813014)

[Github Source Repos for ACE Liveliness Probe (The SoE ACE project) 8](#_Toc813015)

[Description 8](#_Toc813016)

[Testing Liveliness Probe 8](#_Toc813017)

[ACE Integration Micro Service 1 9](#_Toc813018)

[Github Source Repos for ACE Micro Service 1 9](#_Toc813019)

[Description 9](#_Toc813020)

[Testing Micro Service 1 10](#_Toc813021)

[ACE Integration Micro Service 2 11](#_Toc813022)

[Github Source Repos for ACE Micro Service 2 11](#_Toc813023)

[Description 11](#_Toc813024)

[Testing Micro Service 2 12](#_Toc813025)

[Building the Environment on ICP 14](#_Toc813026)

[Pre-Req work – Customization of Microclimate/Jenkins build scripts on ICP 14](#_Toc813027)

[1. Create pipeline deployment namespace 14](#_Toc813028)

[2. Edit ClusterImagePolicy 14](#_Toc813029)

[3. Create Docker Registry secret to microclimate namespace 14](#_Toc813030)

[4. Initialise Helm and login 14](#_Toc813031)

[5. Create Helm secret 14](#_Toc813032)

[6. Create Docker Registry secret for microclimate-pipeline-deployments namespace 14](#_Toc813033)

[7. Update ImagePullSecret for microclimate-pipeline-deployments namespace 14](#_Toc813034)

[8. Customise Jenkins library 14](#_Toc813035)

[9. Deploy Microclimate helm chart 15](#_Toc813036)

[Create a project in Microclimate and Deploy 16](#_Toc813037)

[Troubleshooting 16](#_Toc813038)

[Creating the Microclimate projects and Jenkins pipelines 17](#_Toc813039)

[ACE on ICP Integration Standard Operating Environment Image 17](#_Toc813040)

[ACE on ICP Integration Micro Service 1 23](#_Toc813041)

[ACE on ICP Integration Micro Service 2 24](#_Toc813042)

[ACE on ICP Integration Micro Service Helm Release – Individual services 25](#_Toc813043)

[ACE on ICP Integration Micro Services Application Helm Release – Application Release 26](#_Toc813044)

[Review of what’s been created on ICP 27](#_Toc813045)

[Microclimate Projects 27](#_Toc813046)

[Jenkins pipelines 28](#_Toc813047)

[ICP Image repository images 28](#_Toc813048)

[ICP Helm releases 29](#_Toc813049)

[Exploring the initial helm releases on ICP 30](#_Toc813050)

[Integration Micros Services Application helm release 30](#_Toc813051)

[Explore Integration micro service 2 30](#_Toc813052)

[Explore Integration micro service 1 32](#_Toc813053)

[Individual Integration micros service helm release 35](#_Toc813054)

[Running the Pipelines manually to load the images onto ICP 40](#_Toc813055)

[Building the Image for SoE – GitHub ACE SoE Build 40](#_Toc813056)

[Source Github repository 40](#_Toc813057)

[Microclimate SOE Project 40](#_Toc813058)

[Jenkins Pipeline for SoE 41](#_Toc813059)

[SoE Image on ICP 41](#_Toc813060)

[Deploying the SoE 41](#_Toc813061)

[Building the Image for Micro Service 1 – GitHub ACE Micro Service 1 Build 42](#_Toc813062)

[Source Github repository 42](#_Toc813063)

[Microclimate Micro Service 1 Project 42](#_Toc813064)

[Jenkins Pipeline for Micro Service 1 43](#_Toc813065)

[Micro Service 1 Image on ICP 43](#_Toc813066)

[Building the Image for Micro Service 2 – GitHub ACE Micro Service 2 Build 44](#_Toc813067)

[Source Github repository 44](#_Toc813068)

[Microclimate Micro Service 2 Project 45](#_Toc813069)

[Jenkins Pipeline for Micro Service 2 46](#_Toc813070)

[Micro Service 2 Image on ICP 46](#_Toc813071)

[Deploying Testing and Demoing 48](#_Toc813072)

[Deploying Micro Service 1 and Micro Service 2 together as a Micro Services Application 48](#_Toc813073)

[Github repository - ACEonICPIntMicSrvHelm 48](#_Toc813074)

[Chart Files for Integration Micro Service 1 48](#_Toc813075)

[Sub Chart files for Integration Micro Service 2 49](#_Toc813076)

[YAML Chart file modifications - How it hangs together 50](#_Toc813077)

[Microclimate project aceonicpintmicsrvapplhelm 52](#_Toc813078)

[Jenkins Pipeline - aceonicpintmicsrvapplhelm 52](#_Toc813079)

[Helm Release on ICP for Integration Micro Services Application 54](#_Toc813080)

[Integration Micro Service One 56](#_Toc813081)

[Integration Micro Service Two 57](#_Toc813082)

[Testing the Integration Micro Services Application 59](#_Toc813083)

[Deploying Micro Service 2 on it’s own for testing 63](#_Toc813084)

[Source Github repository - ACEonICPIntMicSrvHelm 63](#_Toc813085)

[Microclimate Project 64](#_Toc813086)

[Jenkins Pipeline 64](#_Toc813087)

[Testing Integration Micro Service 2 standalone 65](#_Toc813088)

[Deploying Micro Service 1 on it’s own for testing 71](#_Toc813089)

[Source Github repository - ACEonICPIntMicSrvHelm 71](#_Toc813090)

[Microclimate Project 73](#_Toc813091)

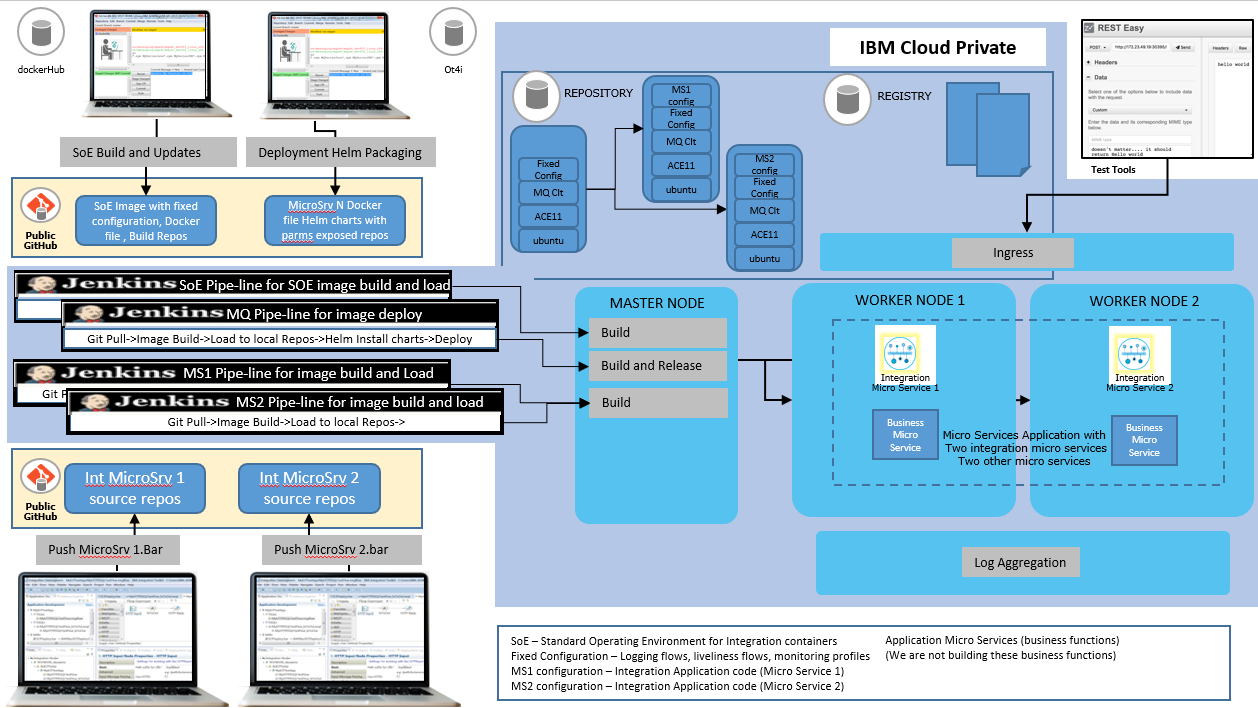
[Jenkins Pipeline 73](#_Toc813092)

[Testing Integration Micro Service 1 standalone 74](#_Toc813093)

[Testing Liveliness Probe 74](#_Toc813094)

# Introduction

## Scenario Overview Diagram



## Overview Description

### The Scenario

The ICP Cloud represents an environment that exists in the standard Software Delivery Life Cycle (could be SIT, QA, Perf, Prod (or potentially one of those environments in an ICP cluster namespace)

We are delivering a Micro Services application consisting of multiple micro services two of which have requirements that are integration centric and ACE has been chosen as the runtime for these functions

Our fictitious organization maintains a standard operating environment (container image) for micro service integration with a fixed configuration component on top of which the integration micro services are added and then the images deployed and run as a helm release.

The primary mechanism for deployment will be Jenkins but we'll use Microclimate as it sorts out security on the ICP instance for us and standup the Jenkins toolchains.

### The Integration Micro Services

The Integration Micro Service 1 will call Integration Micro Service 2

Integration Micro Service 1 RESTInput(HTTP) ->Mapping Node ->RESTRequest (call Integration Microservice 2)->RESTReply(HTTP)

Integration Micro Service 2 RESTInput(HTTP)->Mapping Node Payload+"Hello from Integration Microservice 2"->RESTReply(HTTP)

Integration Micro Service 2 can be called directly

We have set up the integration in this way in order to position for using Istio as a service mesh to route between integration micro services. However, in this initial example I will make use of the ICP DNS service that uses services names to resolve host names and port numbers. So the base URL on the RESTRequest Node in Micro Service 1 will call Micro Service 2 via its service name.

### The docker images

The ACE Standard Operating Environment image is:

Ubuntu

ACEv11.0.0.2

MQ v9.1 Client

Fixed configuration - a bar with RESTful Service, a custom Liveliness Listener

The ACE micro services images build FROM this (base) Standard Operating Environment

### The Personas

Developer 1 and Developer 2 have their own ACE Toolkits and workspaces and are “just” developers. They write ACE services and flows and build BAR files.

Builder(s) are responsible for maintaining the SoE image and building integration microservices images FROM the SoE based on the BAR files created by the Developers

Deployer(s) are responsible for packaging and deployment of micro services application that can include integration micro services. They use Helm and the ICP Devops tooling and pipelines to deploy.

### Customized Jenkins Pipelines

Microclimate you have to customize the jenkins build scripts (micro climate is really set up for Java and NodeJS).

We made changes to the Jenkins pipeline scripts in order to get greater flexibility in the Image Build, Deploy Only and Image Build and Deploy. This allowed us greater control in demoing.

<https://github.com/cloudnativedemo/icp-notes/blob/master/microclimate_notes.md>

# Reference Section

## Summary of Github Repositories

ACE Developer Project source for a custom Liveliness probe that will deploy into the Standard Operating Image on which all other images are based

<https://github.com/DAVEXACOM/ACEonICPIntMicSoE>

ACE Developer Project source for Integration Micro Service 1

<https://github.com/DAVEXACOM/ACEonICPIntMicSrv1>

ACE Developer Project source for Integration Micro Service 2

<https://github.com/DAVEXACOM/ACEonICPIntMicSrv2>

ACE Standard Operating Environment Image Build - The base image build with Liveliness probe that other images are build FROM

<https://github.com/DAVEXACOM/ACEonICPIntStdOpImg>

ACE Micro Service 1 Image Build

<https://github.com/DAVEXACOM/ACEonICPIntMicSrv1Img>

ACE Micro Service 2 Image Build

[https://github.com/DAVEXACOM/ACEonICPIntMicSrv2Img](https://github.com/DAVEXACOM/ACEonICPIntMicSrv1Img)

Helm Charts that can deploy ACE Micro Service 1 or 2 as an individual for testing (modify Jenkins file and Values.yaml)

<https://github.com/DAVEXACOM/ACEonICPIntMicSrvHelm>

Helm Charts that deploy both ACE Micro Service 1 and 2 as a micro services application

[https://github.com/DAVEXACOM/ACEonICPIntMicSrvApplHelm](https://github.com/DAVEXACOM/ACEonICPIntMicSrvHelm)

## Summary of Dockerhub Images

ACE Developer Project source for a custom Liveliness probe that will deploy into the Standard Operating Image on which all other images are based

<https://cloud.docker.com/u/davexacom/repository/docker/davexacom/ace11002mqc91soe>

ACE Developer Project source for Integration Micro Service 1

<https://cloud.docker.com/u/davexacom/repository/docker/davexacom/ace11002mqc91intms1>

ACE Developer Project source for Integration Micro Service 2

https://cloud.docker.com/u/davexacom/repository/docker/davexacom/ace11002mqc91intms2

You can use the dockerHub images with ICP rather than the on board repository OR pull them to you workstations and play around with them in a docker only (non K8s) environment.

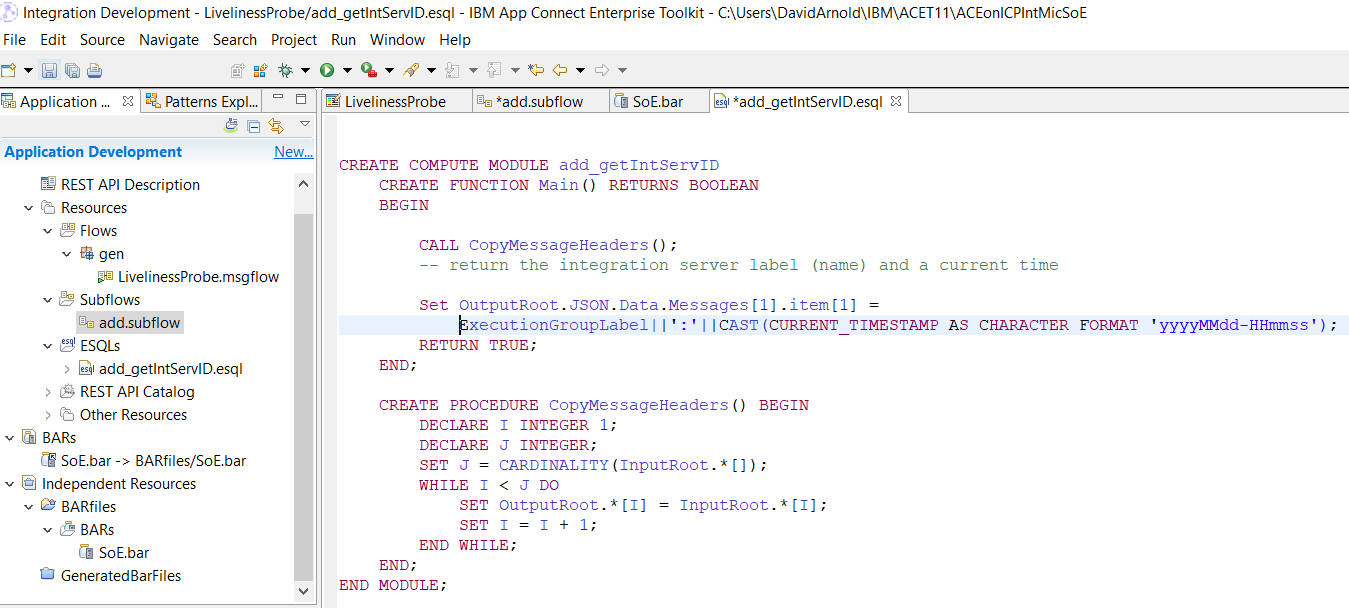
## ACE Integration Liveliness Probe

### Github Source Repos for ACE Liveliness Probe (The SoE ACE project)

<https://github.com/DAVEXACOM/ACEonICPIntMicSoE>

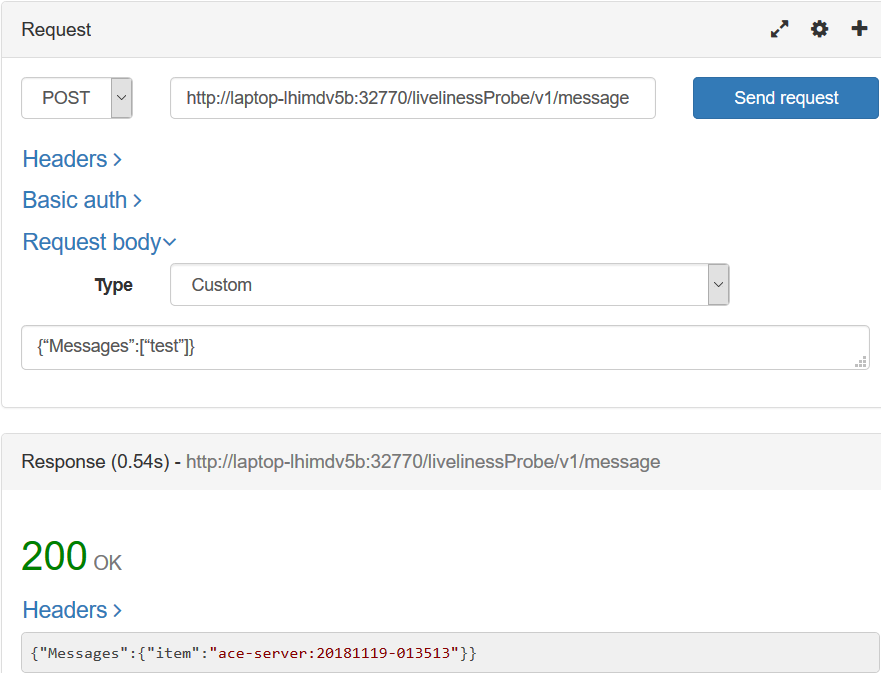
### Description

The Liveliness Probe Service is an other restful service that we deploy into the ACE Standard operating environment. The base image from which Micro Service 1 and Micro Service 2 images are built. So it appears in all ACE Integration Micro Services. (this is not the service baked into the cloud paks we turn those off to demonstrate having a customer centric base standard image. (an SoE).



### Testing Liveliness Probe

Note it returns the integration server name plus a current timestamp for input: {“Messages”:[“test”]}



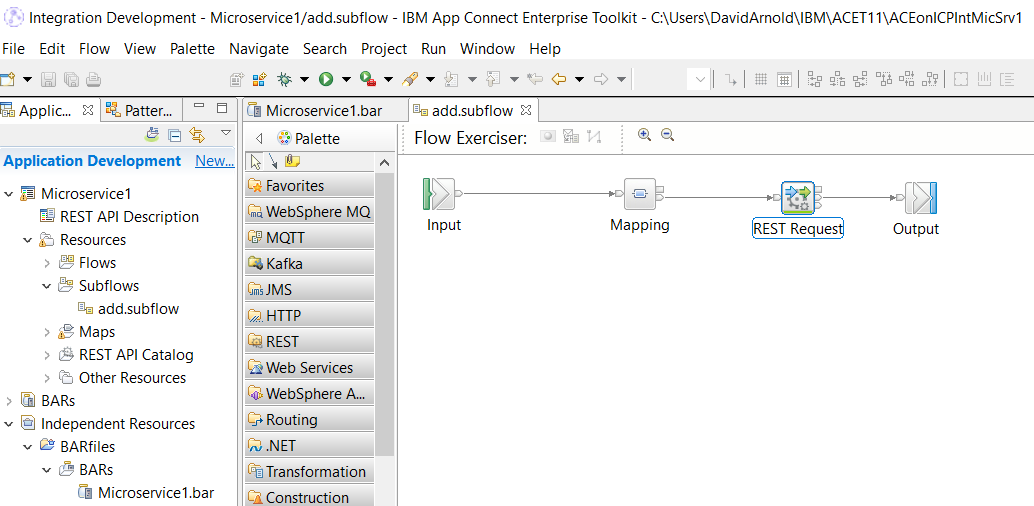
## ACE Integration Micro Service 1

### Github Source Repos for ACE Micro Service 1

https://github.com/DAVEXACOM/ACEonICPIntMicSrv1

### Description

Integration Micro Service 1 calls Integration Micro Service 2 via Rest request



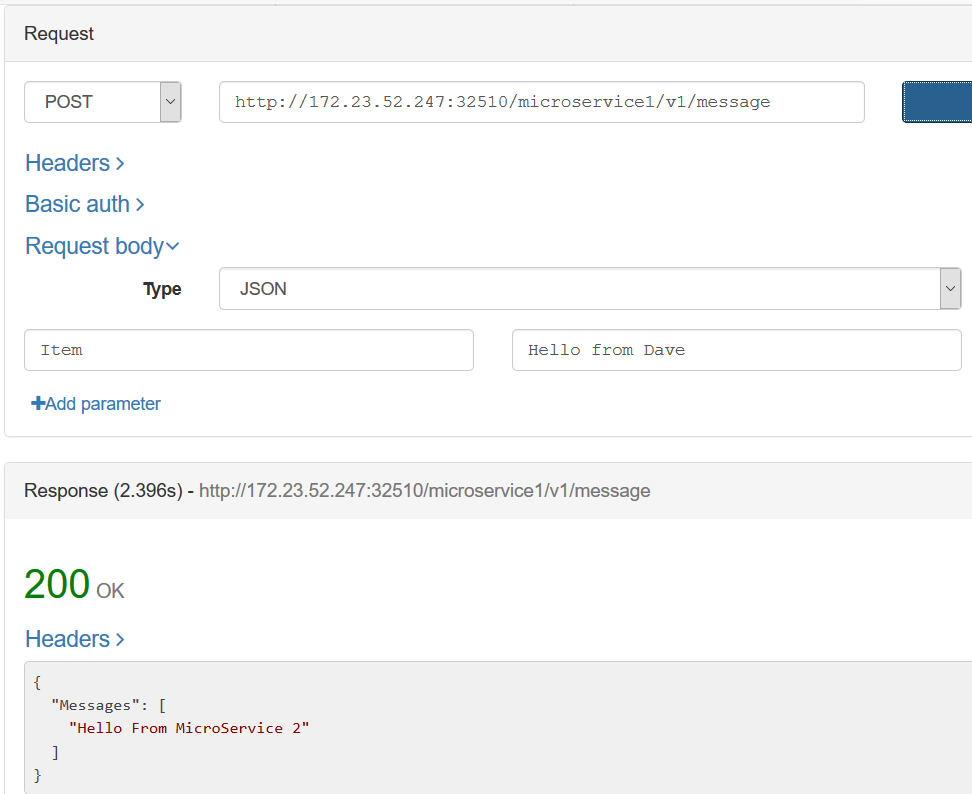
The base URL on the REST Request Node leverages the ICP DNS service naming convention to call MS2 via its service name. This will need to be replaced with Istio service mesh routing.

<http://ace11002mqc91intmsall-intmicsrvtwo:7800/microservice2/v1>

### Testing Micro Service 1

Micro service 1 if deployed and tested standalone will fail.

Micro service 1 if deployed and tested with micro service 2 will return “hello from Micro Service 2”



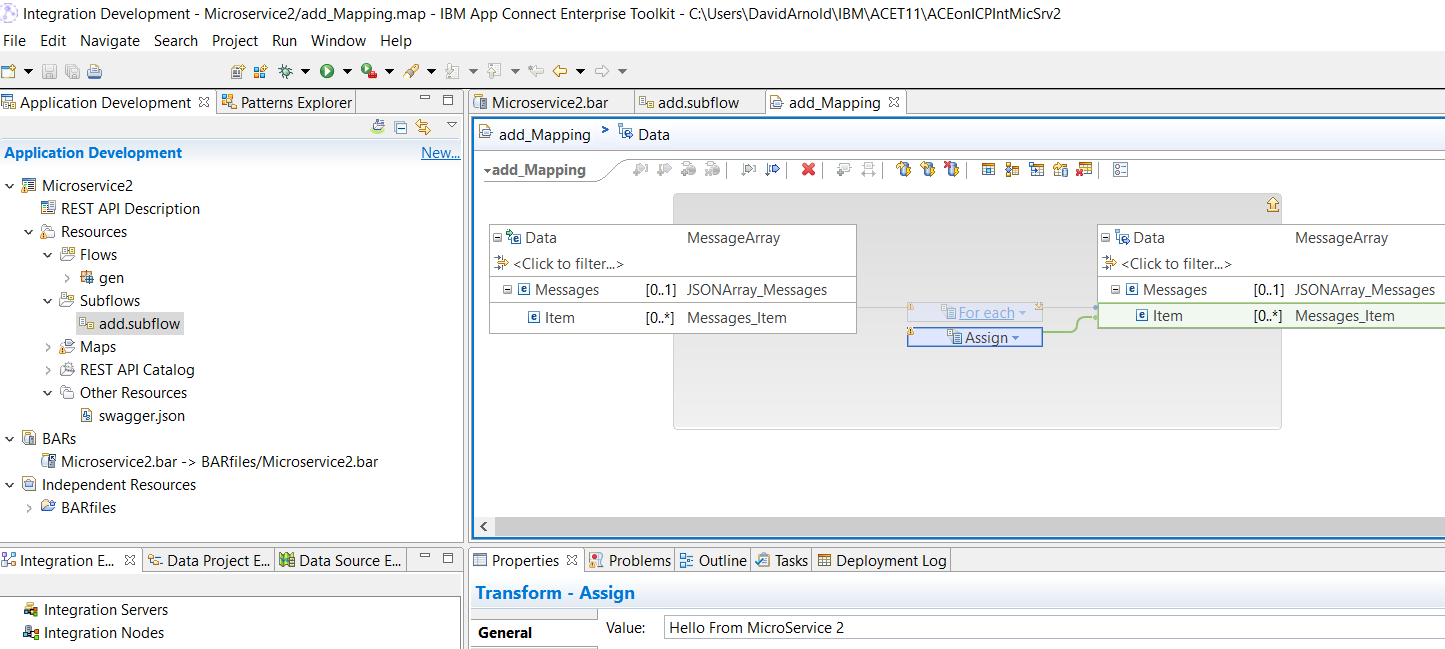
## ACE Integration Micro Service 2

### Github Source Repos for ACE Micro Service 2

https://github.com/DAVEXACOM/ACEonICPIntMicSrv2

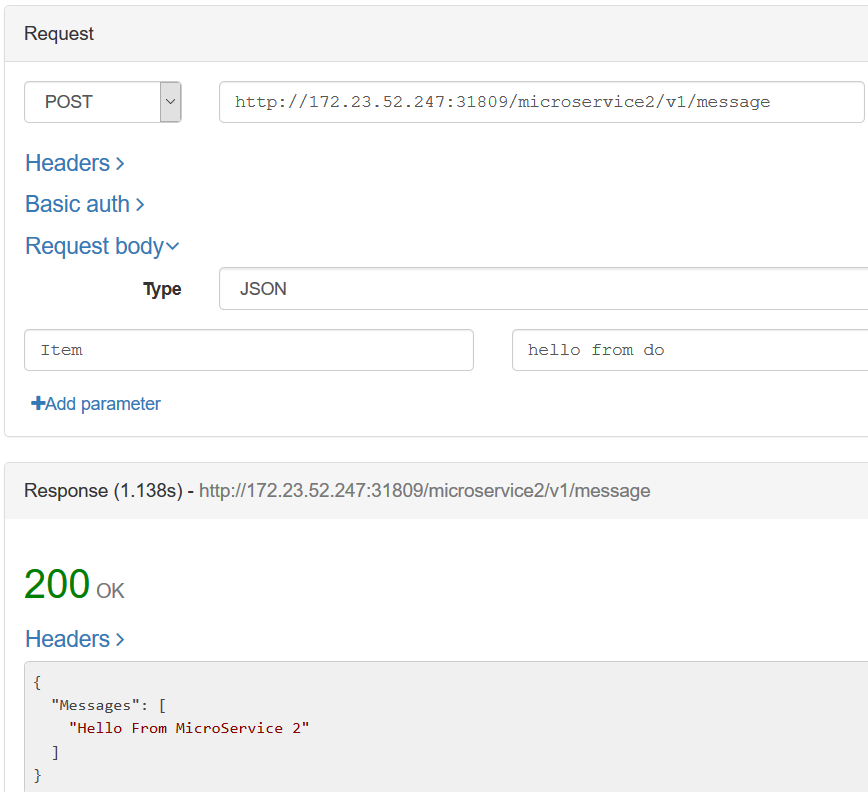
### Description

Integration Micro Service 2 can be called directly as a rest service or access by calling Micro Service 1 (which in turn calls it)



### Testing Micro Service 2

Note it returns hello from Micro Service 2



# Building the Environment on ICP

## Pre-Req work – Customization of Microclimate/Jenkins build scripts on ICP

Documentation is kept up to date on this process here

<https://github.com/cloudnativedemo/icp-notes/blob/master/microclimate_notes.md>

### **1. Create pipeline deployment namespace**

kubectl create namespace microclimate-pipeline-deployments

### **2. Edit ClusterImagePolicy**

kubectl edit clusterimagepolicies ibmcloud-default-cluster-image-policy

To add the following:

- name: mycluster.icp:8500:\*

- name: docker.io/maven:\*

- name: docker.io/lachlanevenson/k8s-helm:\*

- name: docker.io/jenkins/\*

### **3. Create Docker Registry secret to microclimate namespace**

kubectl create secret docker-registry microclimate-registry-secret \

--docker-server=mycluster.icp:8500 \

--docker-username=admin \

--docker-password=admin \

--docker-email=null

### **4. Initialise Helm and login**

helm init --client-only --skip-refresh

cloudctl login -a https://mycluster.icp:8443 -u admin -p admin -c id-mycluster-account -n default --skip-ssl-validation

### **5. Create Helm secret**

export HELM\_HOME=$HOME/.helm

kubectl create secret generic microclimate-helm-secret --from-file=cert.pem=$HELM\_HOME/cert.pem --from-file=ca.pem=$HELM\_HOME/ca.pem --from-file=key.pem=$HELM\_HOME/key.pem

### **6. Create Docker Registry secret for** microclimate-pipeline-deployments **namespace**

kubectl create secret docker-registry microclimate-pipeline-secret \

--docker-server=mycluster.icp:8500 \

--docker-username=admin \

--docker-password=admin \

--docker-email=null \

--namespace=microclimate-pipeline-deployments

### **7. Update ImagePullSecret for** microclimate-pipeline-deployments **namespace**

kubectl patch serviceaccount default --namespace microclimate-pipeline-deployments -p '{"imagePullSecrets": [{"name": "microclimate-pipeline-secret"}]}'

### **8. Customise Jenkins library**

By default, the Jenkins library parameter is pointing to <https://github.com/microclimate-dev2ops/jenkins-library> This Jenkins library was a part of the Microclimate DevOps process. When a pipeline is created within a project in Microclimate, microclimate will create a Jenkins pipeline. The pipeline uses this library to .. 1. Pull the code from github repo . .. 2. Build a Docker image based on a Dockerfile found in the repo . .. 3. Authenticate and push the image into ICP's private registry . .. 4. Notify Microclimate to move to the next stage (e.g. deploy) . .. 5. Microclimate 'helm deploy' the helm chart found in the repo (by default it's under the /chart directory) .

Unfortunately, Microclimate only deploy it's supported project types e.g. Swift, NodeJS, Java/Liberty or Springboot. The easiest way to address this limitation is to fork and update the Jenkins library and inject the 'helm deploy' scriptlet onto step 4 (line 400 of microserviceBuilderPipeline.groovy)

container ('helm') {

echo "Attempting to deploy the test release"

def deployCommand = "helm install ${realChartFolder} --values pipeline.yaml --namespace ${namespace} --name ${helmRelease}"

if (fileExists("chart/overrides.yaml")) {

deployCommand += " --values chart/overrides.yaml"

}

if (helmSecret) {

echo "Adding --tls to your deploy command"

deployCommand += helmTlsOptions

}

testDeployAttempt = sh(script: "${deployCommand} > deploy\_attempt.txt", returnStatus: true)

if (testDeployAttempt != 0) {

echo "Warning, did not deploy the test release into the test namespace successfully, error code is: ${testDeployAttempt}"

echo "This build will be marked as a failure: halting after the deletion of the test namespace."

}

printFromFile("deploy\_attempt.txt")

}

* ***Note:*** in my deployCommand, I've created one new variable ${helmRelease}. The variable is defined on the top of the script (line 56 of the microserviceBuilderPipeline.groovy). Alternatively, you can just reuse ${image} as your helm release name .

def helmRelease = (config.releaseName ?: config.image ?: "").trim()

* My forked updated Jenkins library repo can be found [here](https://github.com/cloudnativedemo/jenkins-library) .

### **9. Deploy Microclimate helm chart**

#### **Via Helm command line**

* **Add ibm-charts Helm repo**

helm repo add ibm-charts https://raw.githubusercontent.com/IBM/charts/master/repo/stable/

* **Deploy microclimate Helm chart**

helm install --name microclimate --namespace <target namespace> --set global.rbac.serviceAccountName=micro-sa,jenkins.rbac.serviceAccountName=pipeline-sa,global.ingressDomain=172.23.52.247.nip.io,jenkins.Pipeline.Template.RepositoryUrl=https://github.com/cloudnativedemo/jenkins-library.git,jenkins.Pipeline.Template.Version=master ibm-charts/ibm-microclimate --tls

***Note:*** Replace <172.23.52.247> with your <PROXY\_IP>

#### **Via ICP catalog**

* Select ibm-microclimate from ICP catalog > click Configure
* Provide values for the following parameters:
  + Helm release name: your-microclimate-release-name
  + Namespace: default (or your preferred namespace)
  + Microclimate hostname: microclimate.172.23.52.247.nip.io (replace with your <microclimate.PROXY\_IP.nip.io> or your own hostname)
  + Ensure that you've already created Persistent Volumes for Microclimate and Jenkins
  + Service account name for Portal: micro-sa
  + Jenkins library repository: <https://github.com/cloudnativedemo/jenkins-library.git>
  + Jenkins hostname: jenkins.172.23.52.247.nip.io (replace with your <jenkins.PROXY\_IP.nip.io> or your own hostname)
  + Service account name: pipeline-sa
  + Click deploy

### Create a project in Microclimate and Deploy

Once the Microclimate helm deployment completed, you can start to deploy your custom project

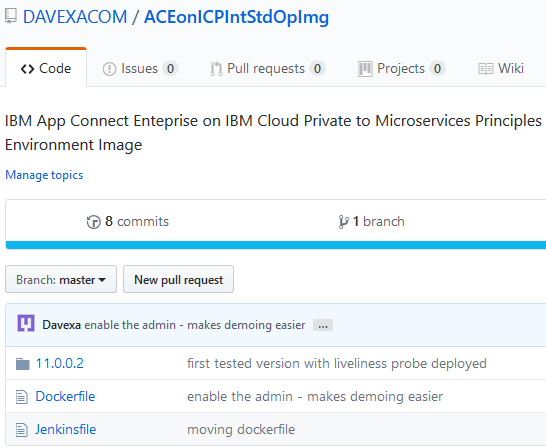
1. Make sure that your project contains a Dockerfile, Jenkinsfile and a chart directory (for helm chart)
2. Launch Microclimate (<https://microclimate.172.23.52.247.nip.io> - replace with your own microclimate hostname) and accept licensing agreement (for first launch only)
3. Select Projects > Click New Project
4. Select Java project type and provide a project name > click Next
5. Select Microprofile/J2EE and keep default value for Context root > click Create
6. Once the project is created, select Pipeline on the left menu
7. Click Create pipeline, and provide name and github repo of the pipeline > click Create pipeline to create a Jenkins pipeline
8. Switch to Jenkins (<https://jenkins.172.23.52.247.nip.io> - replace with your Jenkins hostname) to see if the pipeline has been created and built (refer to the troubleshooting section below if you have to wait for too long)

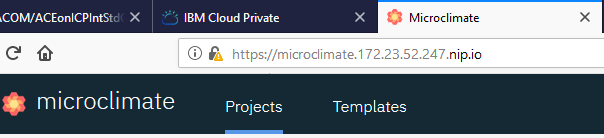
### Troubleshooting

* When your Jenkins pipeline keeps looking for an executor for too long, there's probably an error occurred within your Jenkins containers. To identify the issue:
  + Identify the Jenkins pod name: kubectl get pods -n <NAME\_SPACE\_WHERE\_JENKINS\_INSTALLED> | grep jenkins
  + View the log: kubectl log -n <NAME\_SPACE\_WHERE\_JENKINS\_INSTALLED> <JENKINS\_POD\_NAME> -f
  + Most of the case I found caused by cluster image policy is not defined, you might need to update the default clusterimagepolicy
    - kubectl edit clusterimagepolicies ibmcloud-default-cluster-image-policy

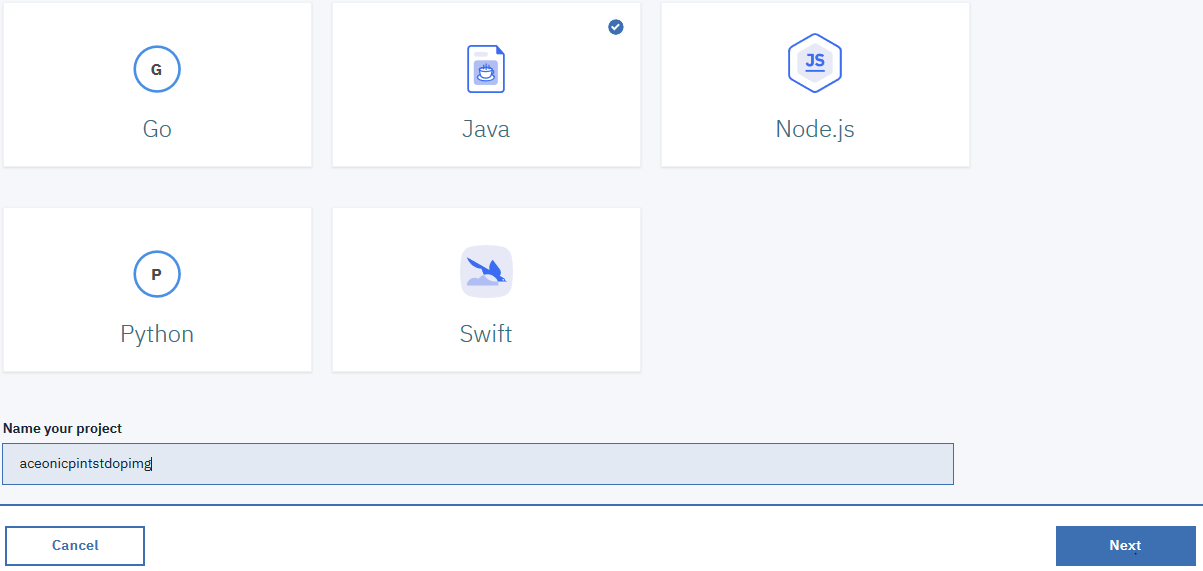
## Creating the Microclimate projects and Jenkins pipelines

### ACE on ICP Integration Standard Operating Environment Image

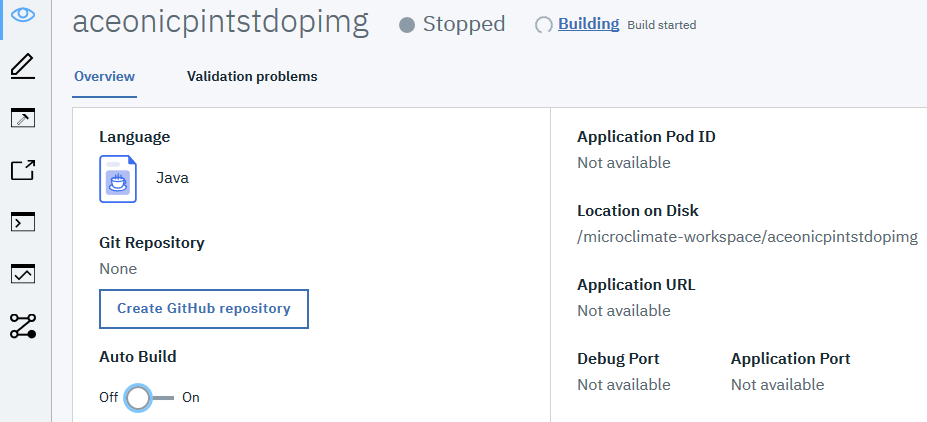








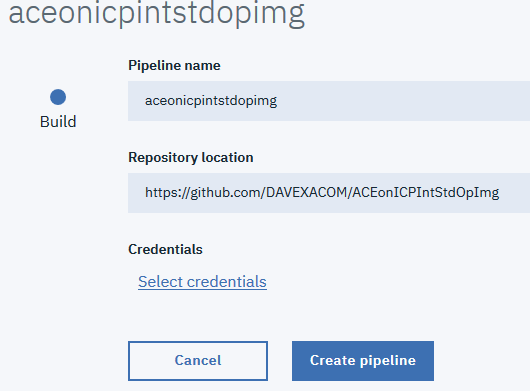
 

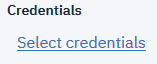


Click on the Pipeline icon

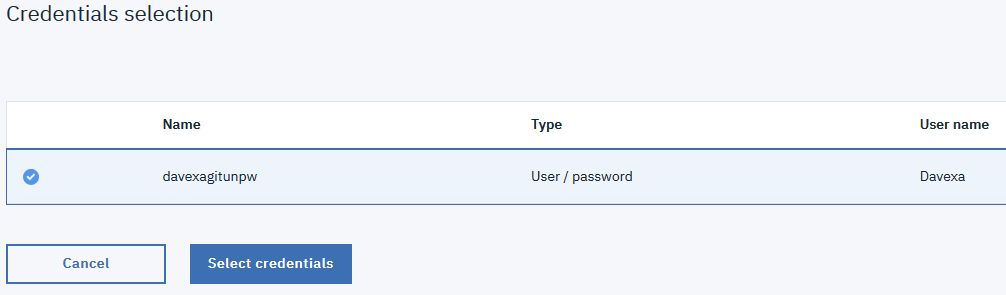




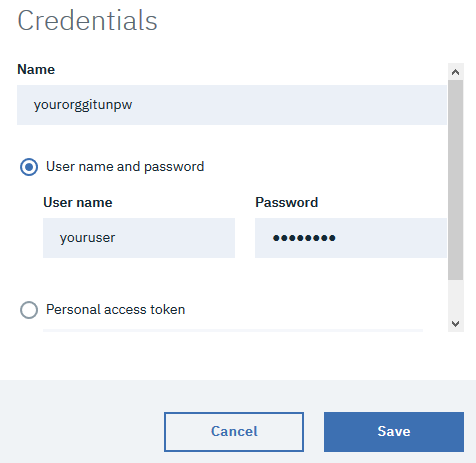




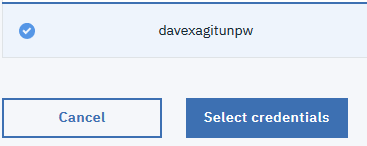
If you already have credentials to your Github organization created use them



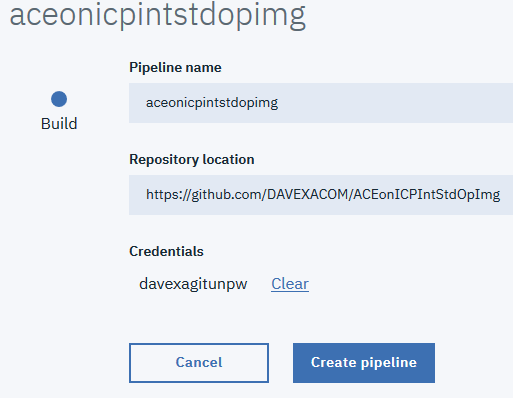
If not click the Add and enter your credentials and save



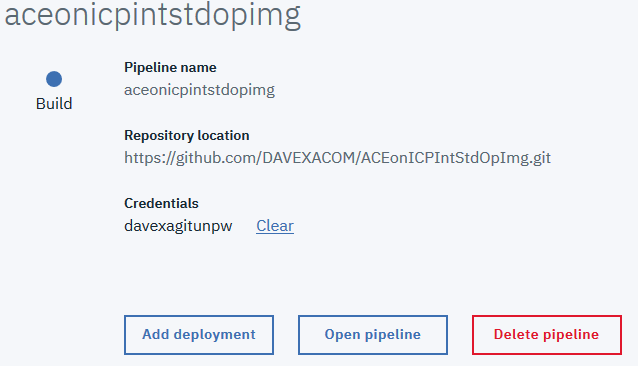
Highlight the credentials and hit select



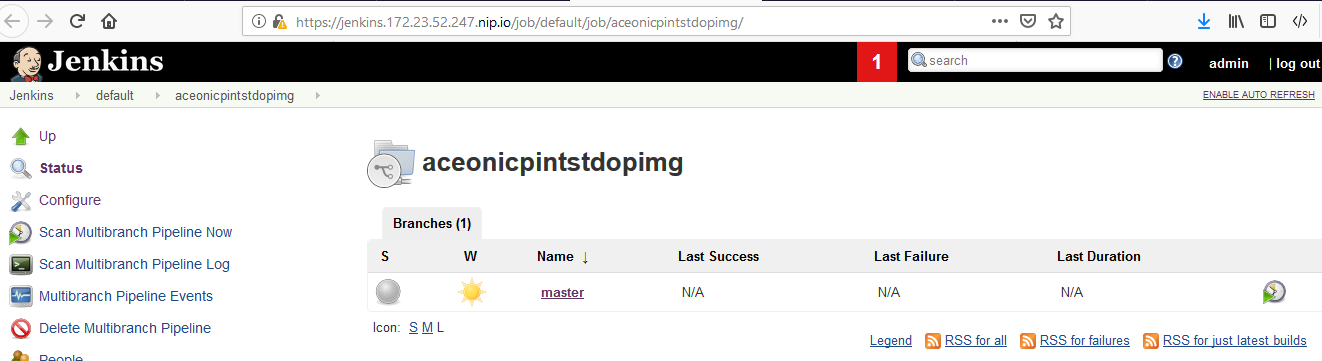
Next create the Jenkins pipeline



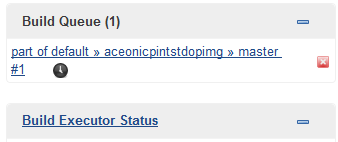
And then Open the pipeline



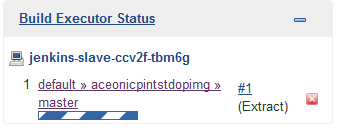
Jenkins will open



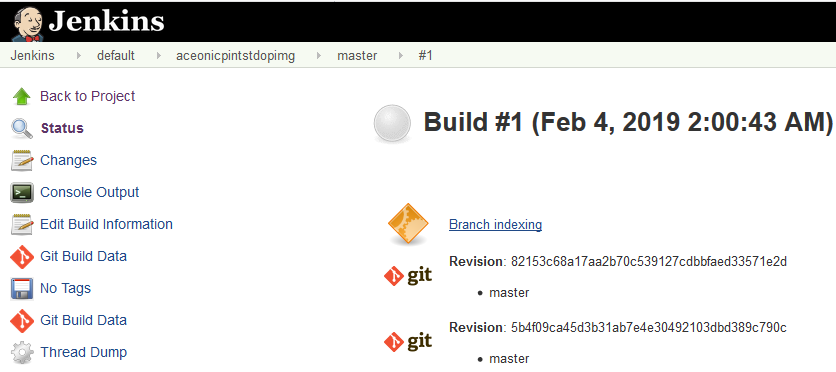
And you’ll see an initial build is initiated



Click on #1



And review the console log to see all is underway and working





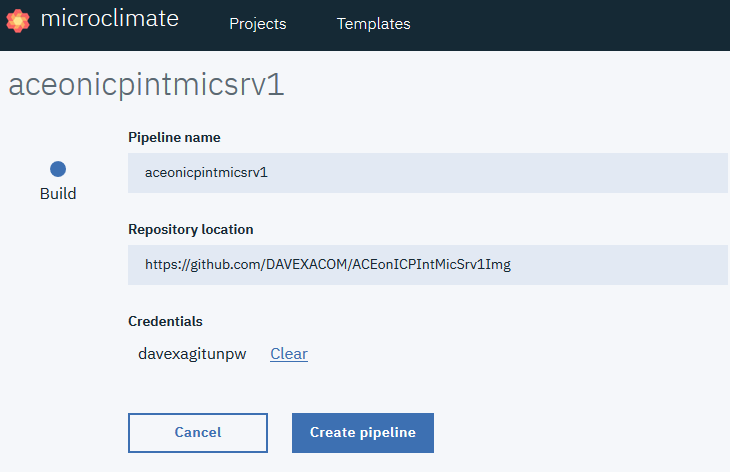
### ACE on ICP Integration Micro Service 1

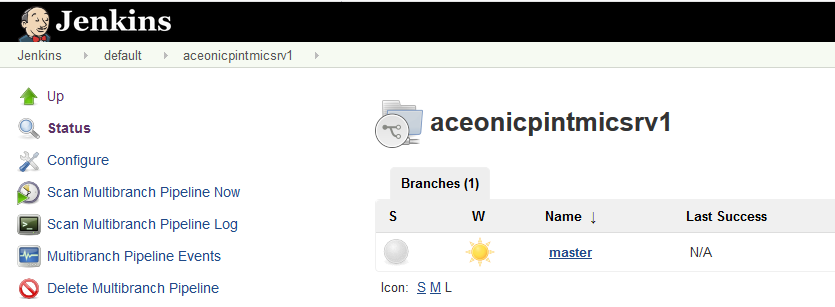
Now you must repeat the process for the integration micro service 1 image using the following:

Naming convention : aceonicpintmicsrv1

Github repository: <https://github.com/DAVEXACOM/ACEonICPIntMicSrv1Img>

This is the image build pipeline for Integration Microservice 1.





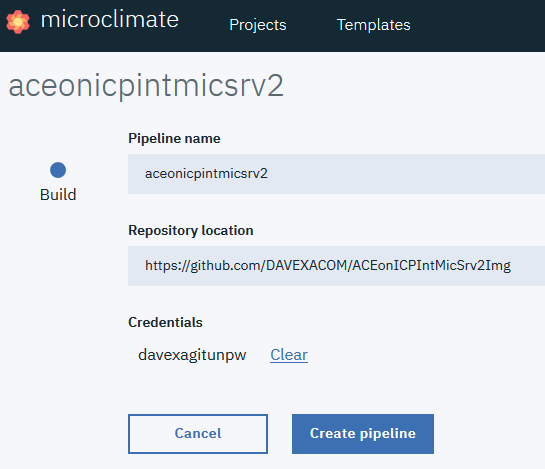
### ACE on ICP Integration Micro Service 2

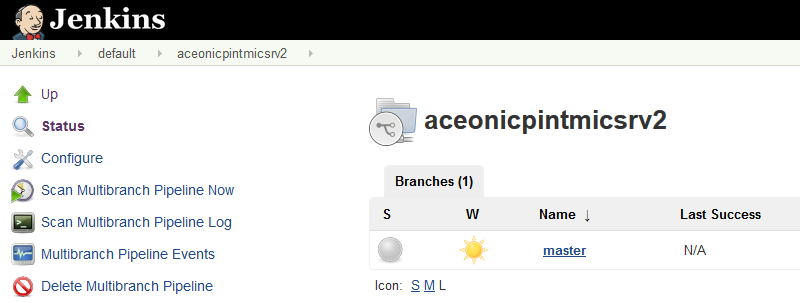
Now you must repeat the process for the integration micro service 2 image using the following:

Naming convention : aceonicpintmicsrv2

Github repository: <https://github.com/DAVEXACOM/ACEonICPIntMicSrv2Img>

This is the image build pipeline for Integration Microservice 2.





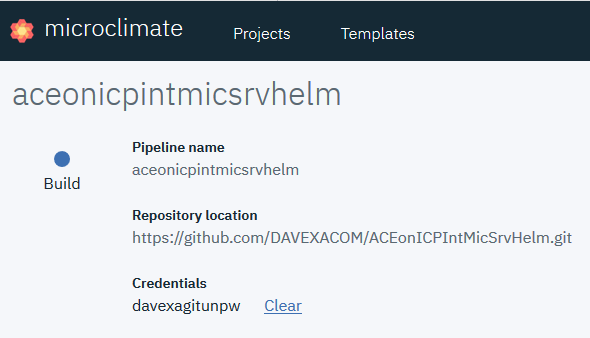
### ACE on ICP Integration Micro Service Helm Release – Individual services

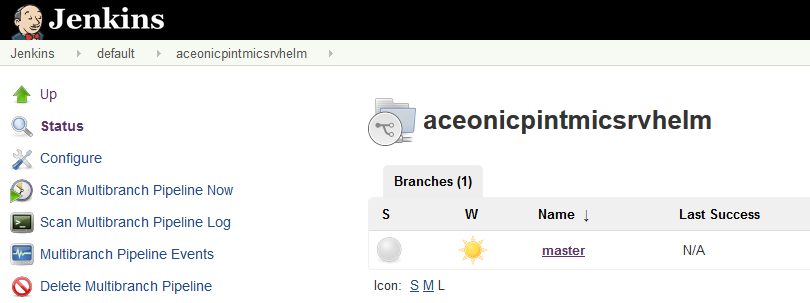
Now you must repeat the process for the integration micro service helm release pipeline for individual services using the following:

Naming convention : aceonicpintmicsrvhelm

Github repository: <https://github.com/DAVEXACOM/ACEonICPIntMicSrvHelm>

This is the Helm Release pipeline that will allow you to deploy either Integration Microservice 1 OR Integration Microservice 2 individually for unit test purposes.





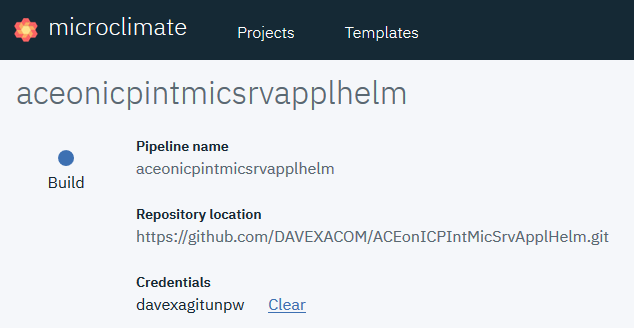
### ACE on ICP Integration Micro Services Application Helm Release – Application Release

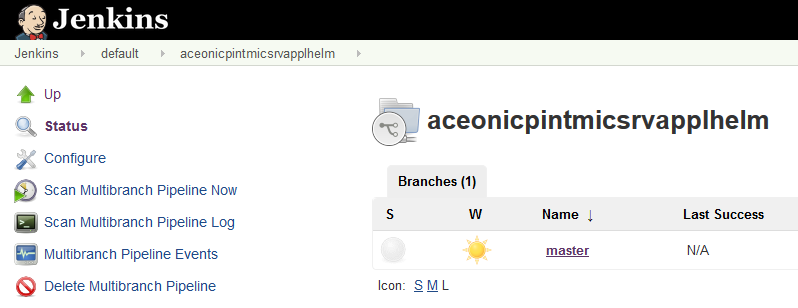
Now you must repeat the process for the integration micro services application helm release pipeline to deploy all services as an application using the following:

Naming convention : aceonicpintmicsrvapplhelm

Github repository: https://github.com/DAVEXACOM/ACEonICPIntMicSrvApplHelm

This is the Helm Release pipeline that will allow you to deploy both Integration Microservice 1 AND Integration Microservice 2 as an application.

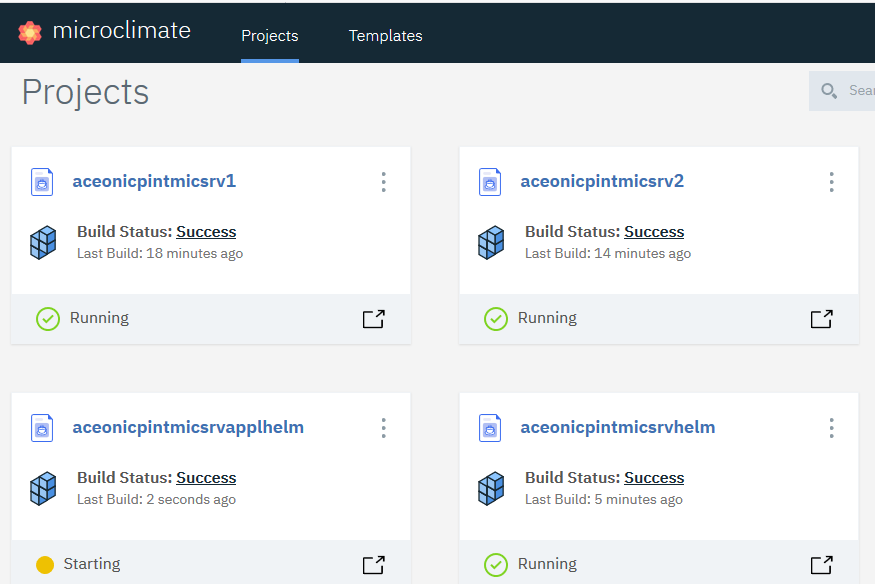


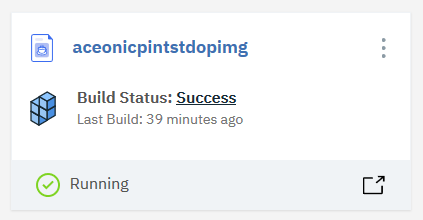


## Review of what’s been created on ICP

### Microclimate Projects

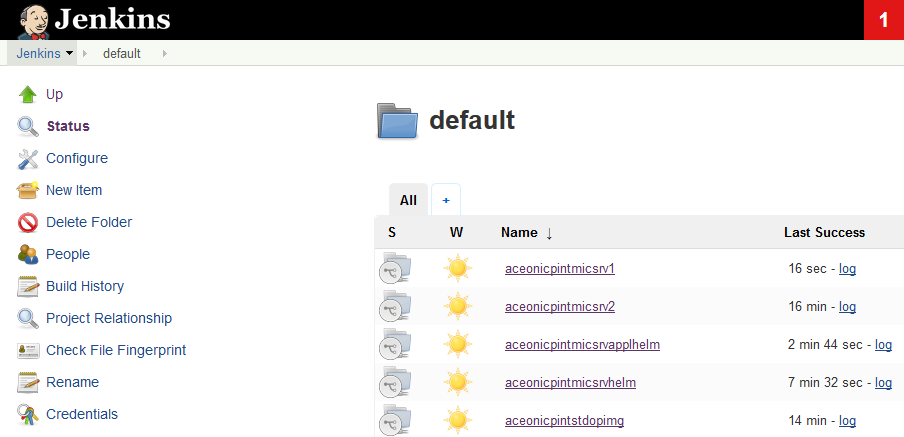
You should now have 5 Microclimate projects



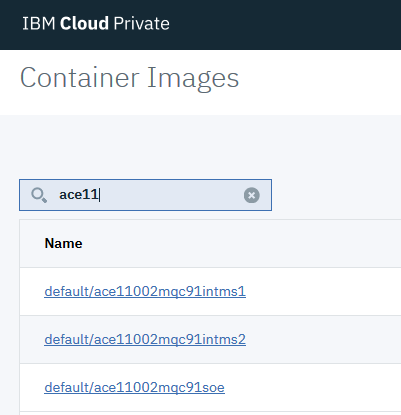


### Jenkins pipelines

And five Jenkins pipelines



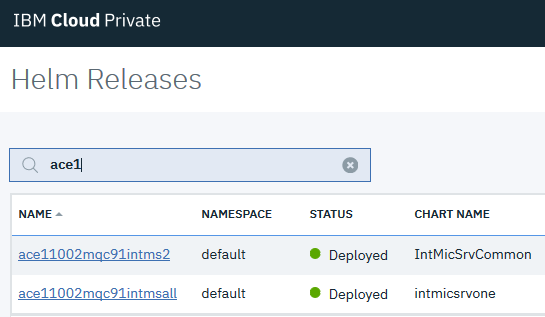
### ICP Image repository images



### ICP Helm releases

The main purpose of the environment creation step was to get the images onto ICP. However,

We can see that in creating to two Helm release pipelines, they have been executed and the release deployed.



# Exploring the initial helm releases on ICP

## Integration Micros Services Application helm release

From the ICP console select Helm Releases

Filter on ace and select the Helm release that deploys multiple integration micro services

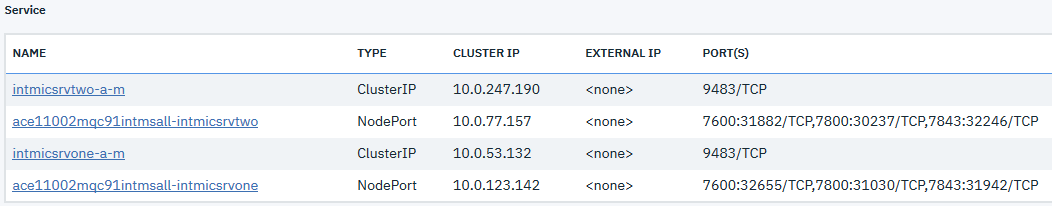


Note the 2 Integration micro services are deployed and each is highly available with 3 instances



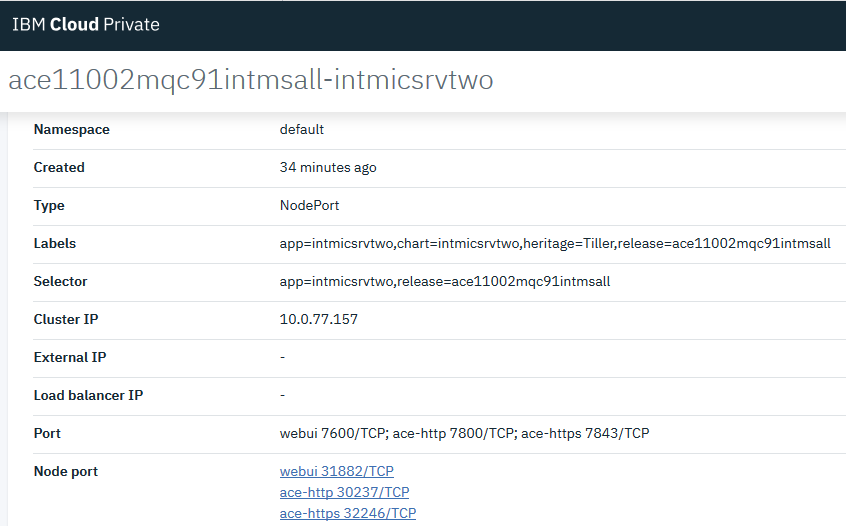
### Explore Integration micro service 2

Scroll down to the services and select integration micro service two

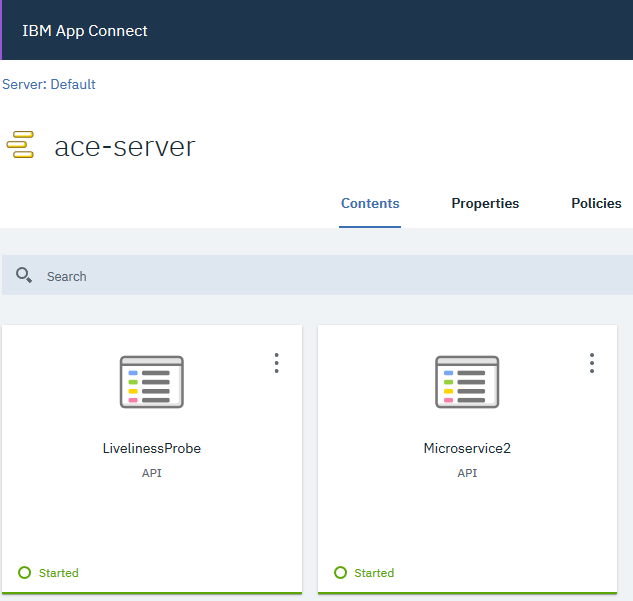


Click on the link





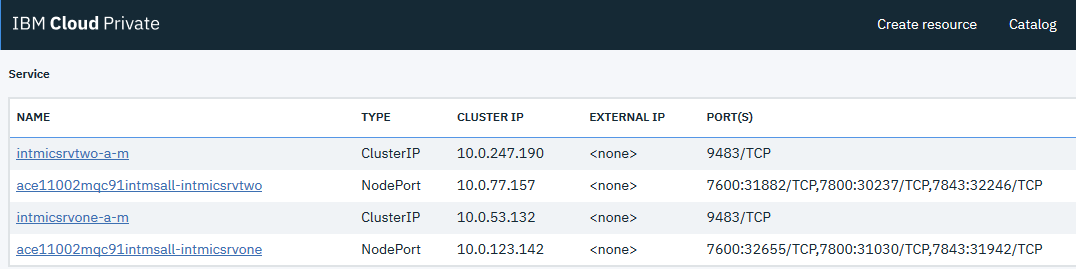
Select the WebUI link to bring up the ACE WebUI



Note Micro service two has its Microservice2 Application API service deployed and the Standard operating environments LivelinessProbe application.

Now go back to the services list for this Helm Releases

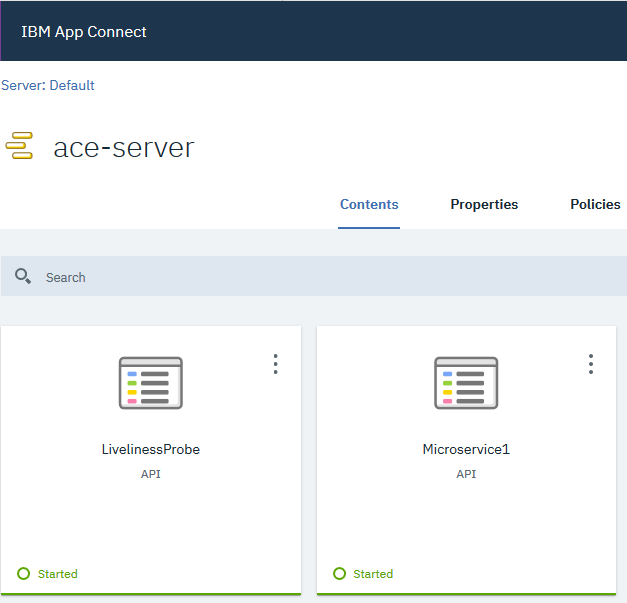
### Explore Integration micro service 1



and follow the above steps to check out micro service 1 following the service link



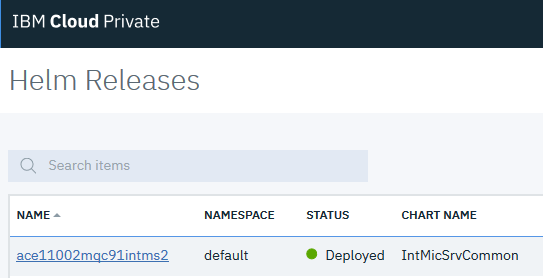
Bring up the ACE WebUI



Note Micro service two has its Microservice2 Application API service deployed and the Standard operating environments LivelinessProbe application.

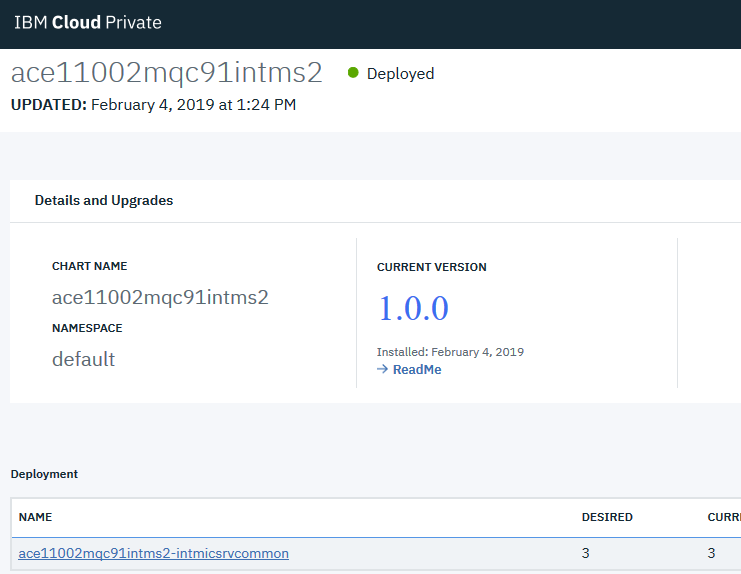
## Individual Integration micros service helm release

From ICP Console select Workloads->Helm Releases

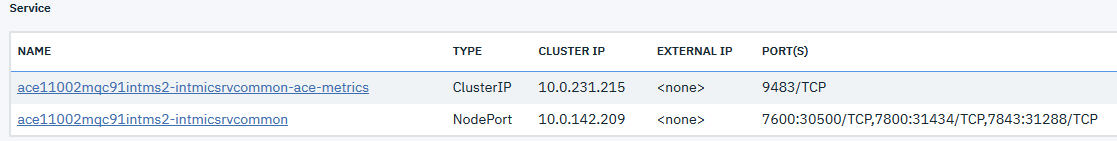


Filter on ace and select



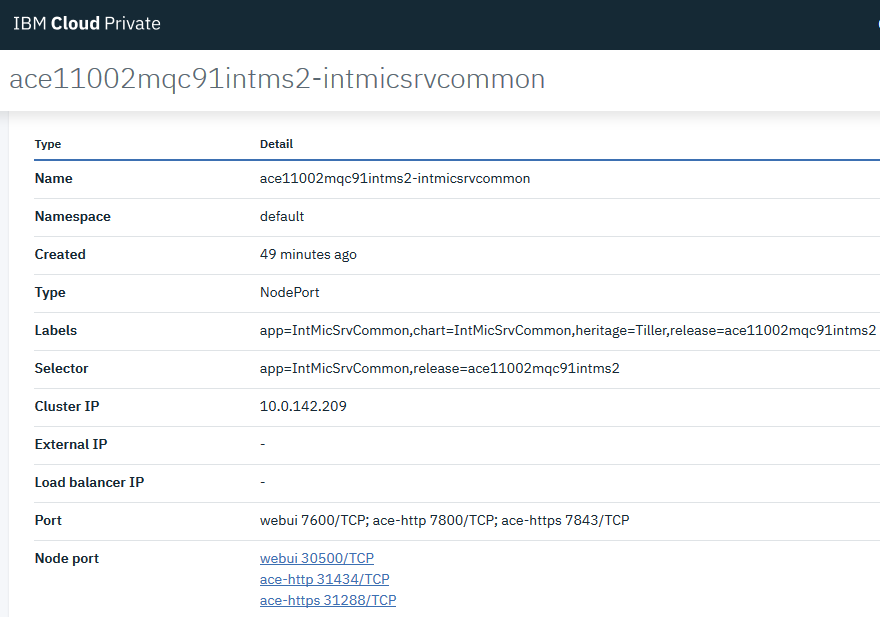


Scroll down to the services

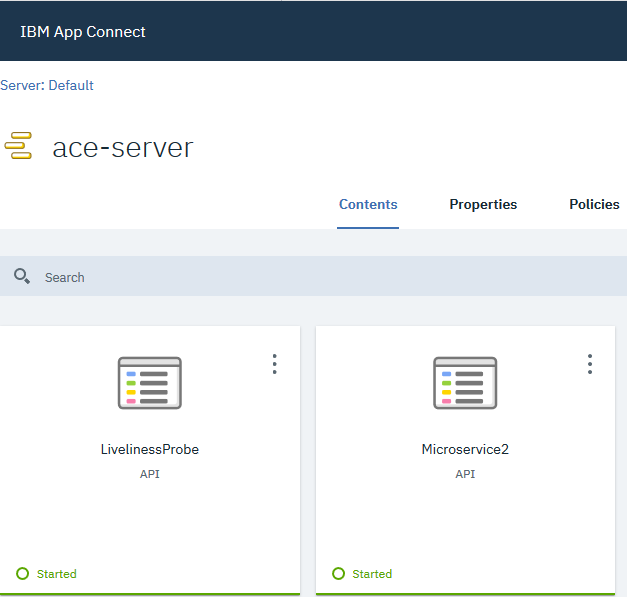


Select integration micro service 2 link





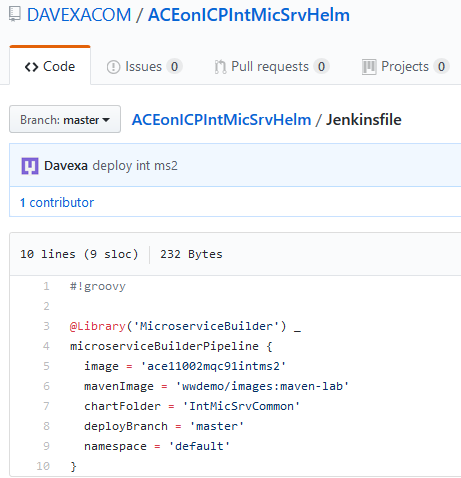
Select the webUI link to bring up the ACE WebUI



Note Micro service two has its Microservice2 Application API service deployed and the Standard operating environments LivelinessProbe application.

The individual microservice2 was deployed because at the time the initial creation of the artifacts on ICP via the Jenkins pipeline referring to <https://github.com/DAVEXACOM/ACEonICPIntMicSrvHelm>

The Jenkins file was set to



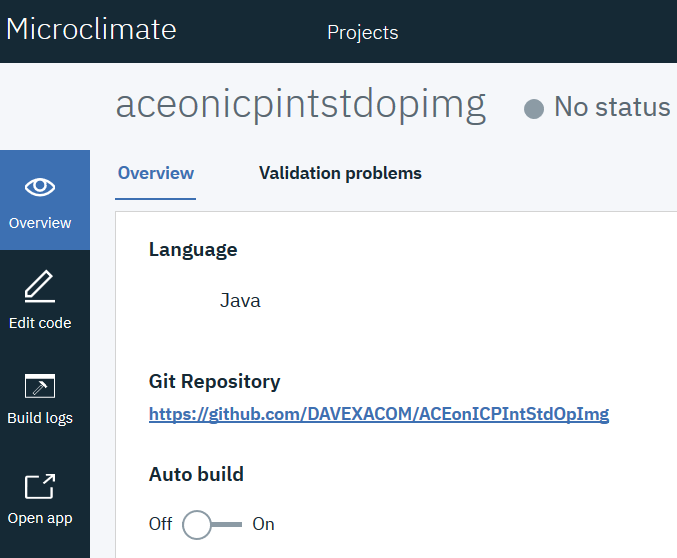
# Running the Pipelines manually to load the images onto ICP

## Building the Image for SoE – GitHub ACE SoE Build

### Source Github repository

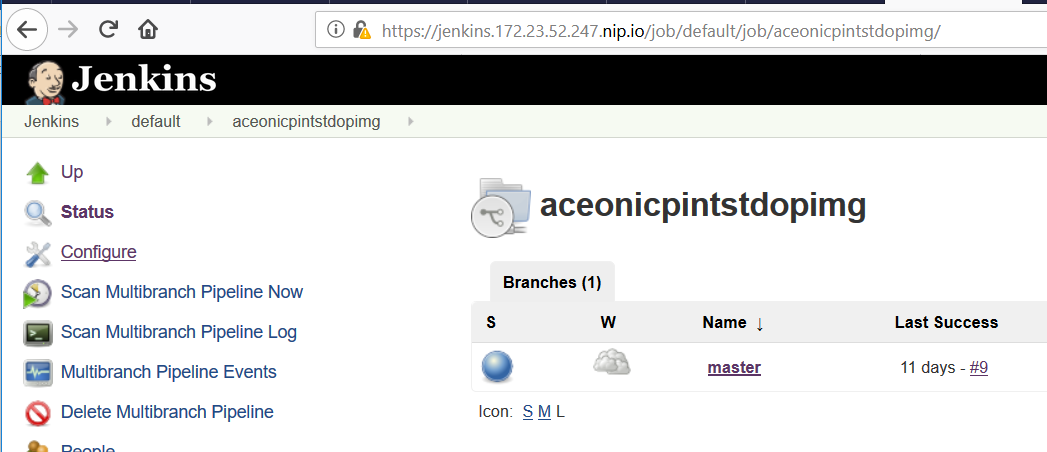
<https://github.com/DAVEXACOM/ACEonICPIntStdOpImg>

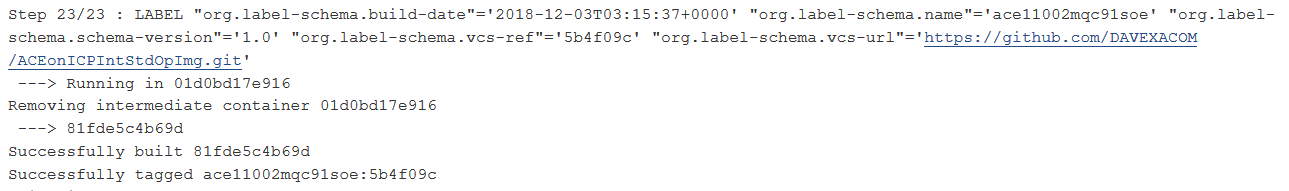
### Microclimate SOE Project



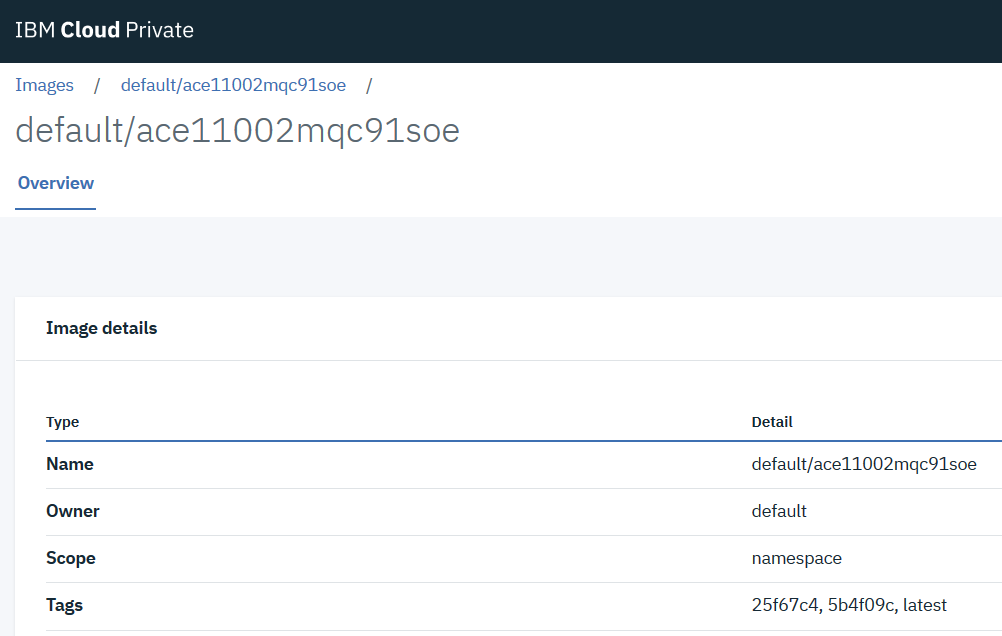


### Jenkins Pipeline for SoE





### SoE Image on ICP



### Deploying the SoE

The SoE Image is never deployed it is the image that Micro Service 1 and 2 are build FROM.

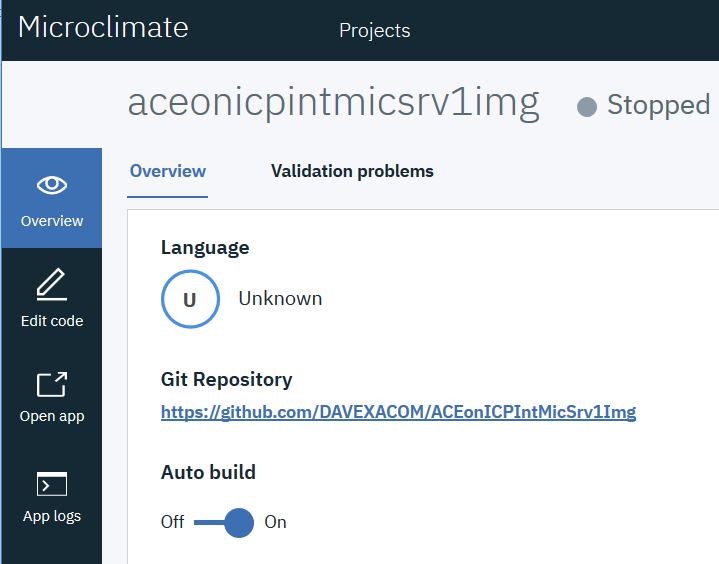
## Building the Image for Micro Service 1 – GitHub ACE Micro Service 1 Build

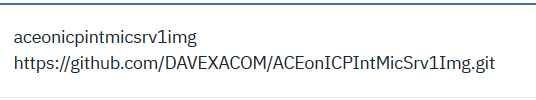
### Source Github repository

<https://github.com/DAVEXACOM/ACEonICPIntMicSrv1Img>

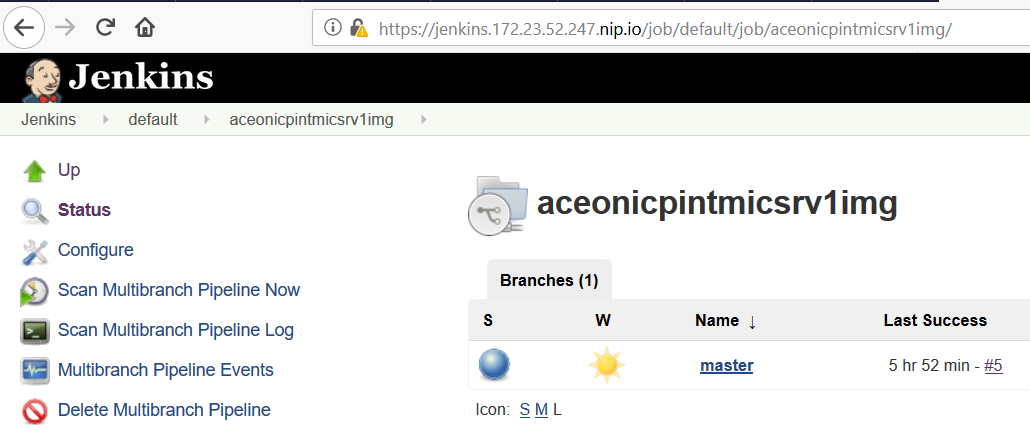
You need to copy the MicroService1.bar from the ACE Micro Service 1 source repository in the <https://github.com/DAVEXACOM/ACEonICPIntMicSrv1Img/tree/master/acesoe/binary> directory

### Microclimate Micro Service 1 Project





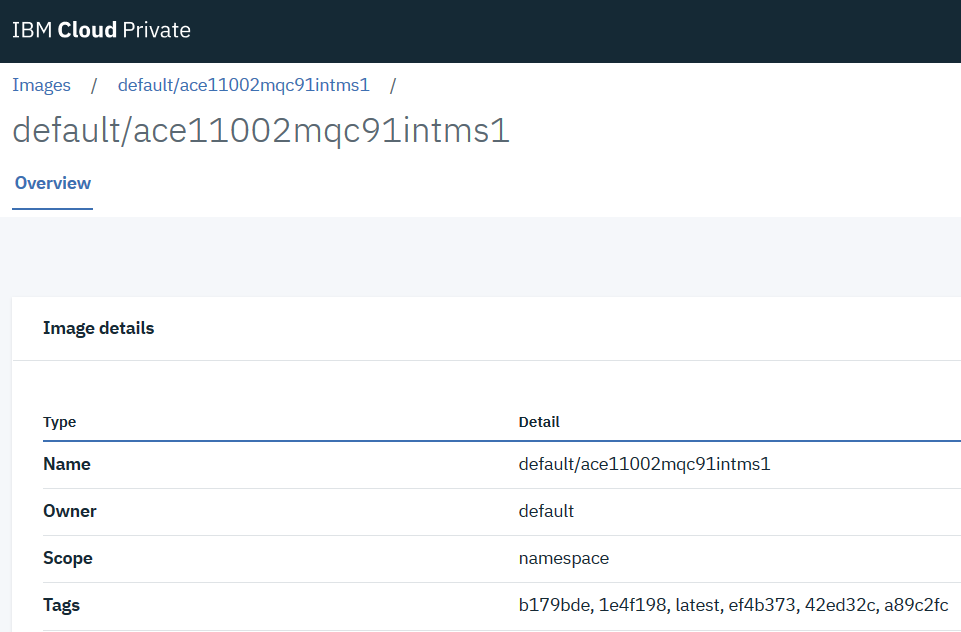
### Jenkins Pipeline for Micro Service 1





Note: The pipeline will try and deploy but fail. We don’t want this pipeline doing the deploy. This is an image build only

### Micro Service 1 Image on ICP



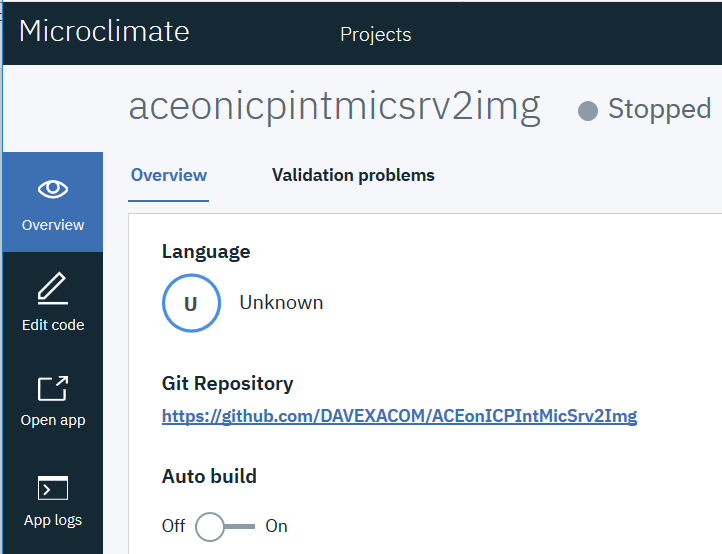
## Building the Image for Micro Service 2 – GitHub ACE Micro Service 2 Build

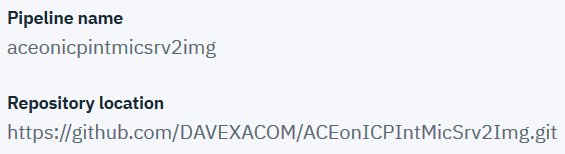
### Source Github repository

<https://github.com/DAVEXACOM/ACEonICPIntMicSrv2Img>

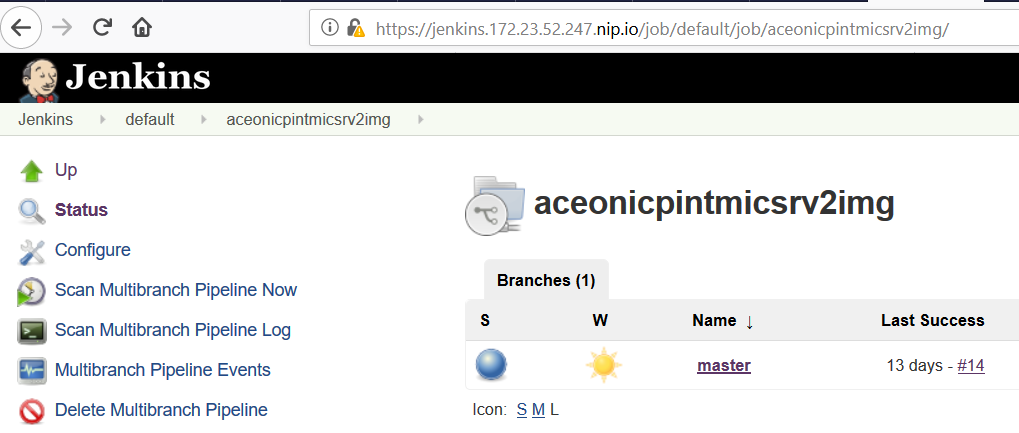
You need to copy the MicroService2.bar from the ACE Micro Service 2 source repository into the <https://github.com/DAVEXACOM/ACEonICPIntMicSrv2Img/tree/master/acesoe/binary> directory

### Microclimate Micro Service 2 Project





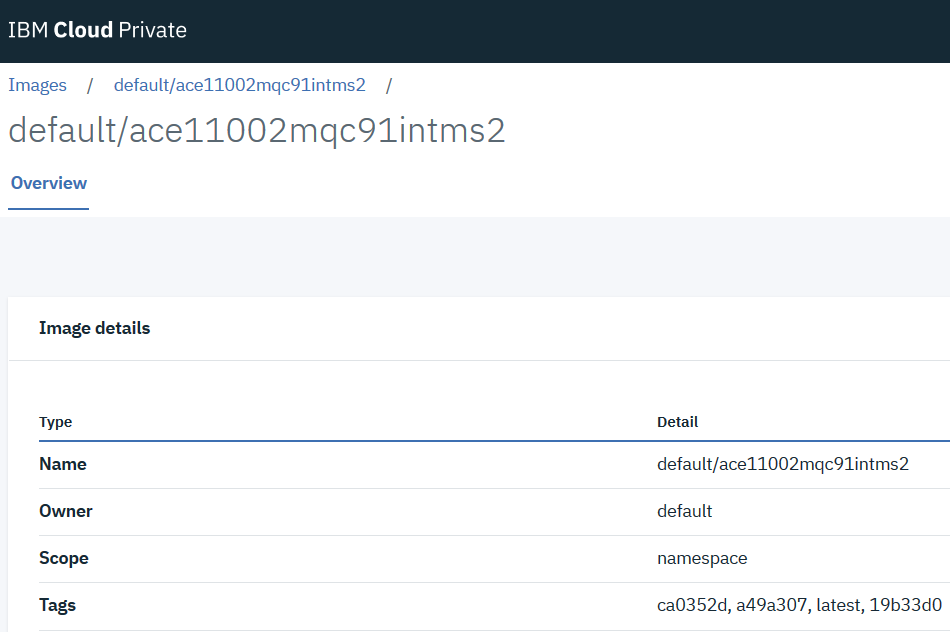
### Jenkins Pipeline for Micro Service 2





Note: The pipeline will try and deploy but fail. We don’t want this pipeline doing the deploy. This is an image build only

### Micro Service 2 Image on ICP

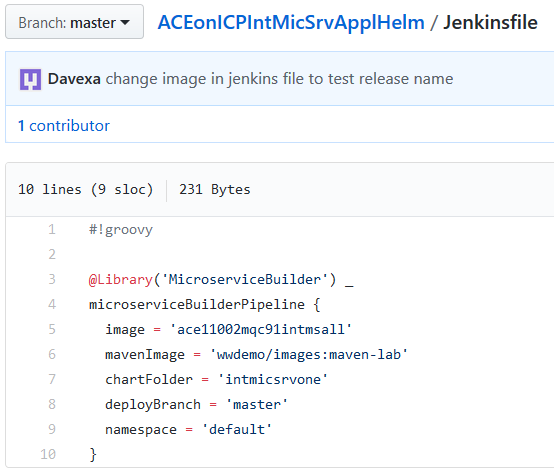


# Deploying Testing and Demoing

## Deploying Micro Service 1 and Micro Service 2 together as a Micro Services Application

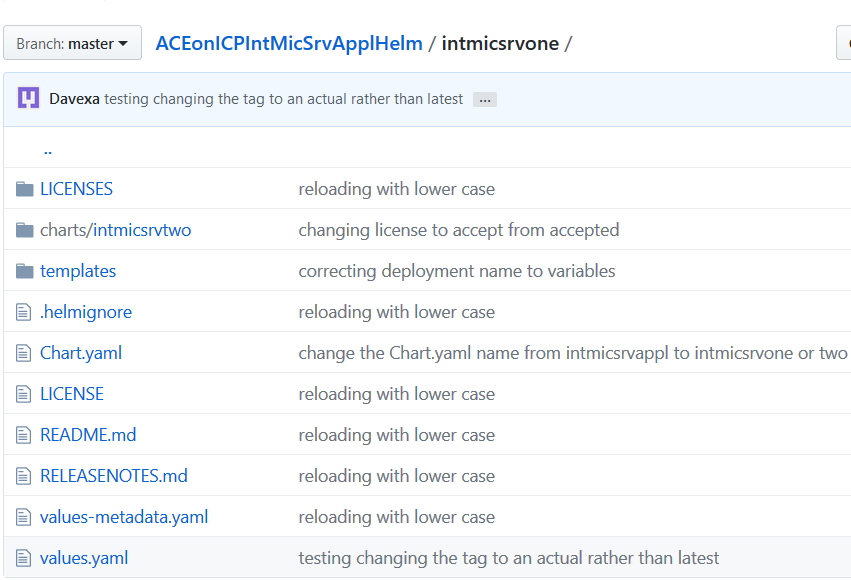
### Github repository - ACEonICPIntMicSrvHelm

<https://github.com/DAVEXACOM/ACEonICPIntMicSrvApplHelm>



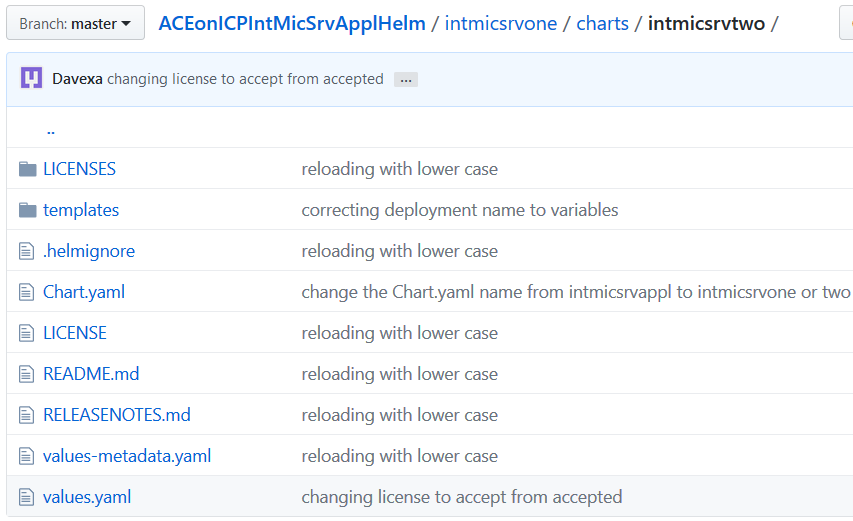
### Chart Files for Integration Micro Service 1

<https://github.com/DAVEXACOM/ACEonICPIntMicSrvApplHelm/tree/master/intmicsrvone>



### Sub Chart files for Integration Micro Service 2

<https://github.com/DAVEXACOM/ACEonICPIntMicSrvApplHelm/tree/master/intmicsrvone/charts/intmicsrvtwo>



### YAML Chart file modifications - How it hangs together

The following YAMl files have been modified from the OT4i content. The main modifications are in the templates YAMLs. Because the sub charts folder and content for integration service two is a copy of main charts files we need to ensure that the secrets and services etc are not created with the same names.

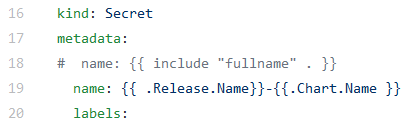
Chart – Chart.yaml



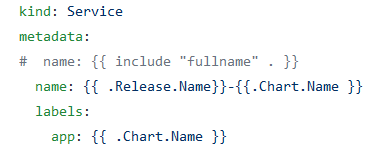
Sub chart – Chart.yaml



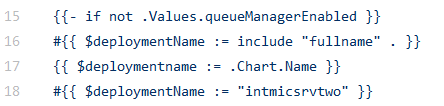
Secrets.yaml for example



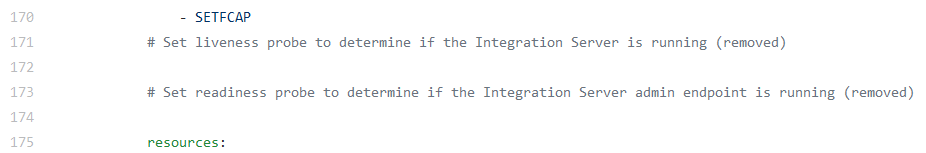
Service.yaml



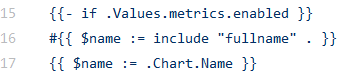
Deployment.yaml



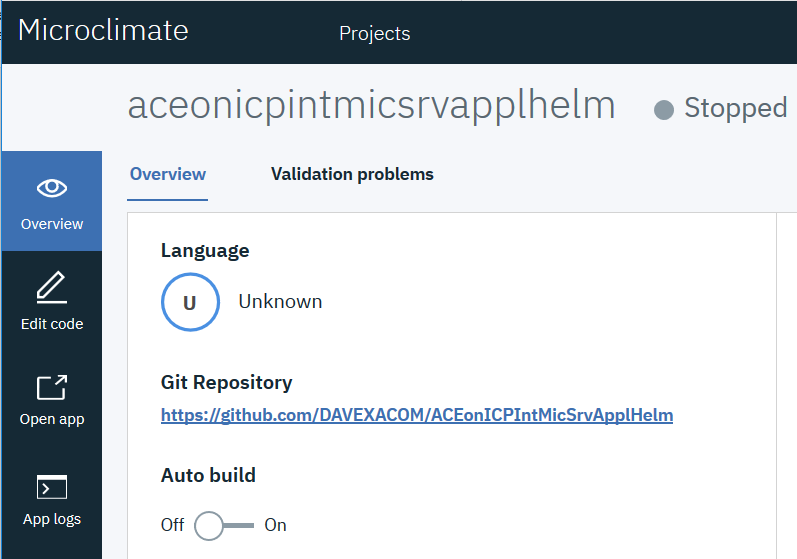
Also removed the cloud pack liveness and readiness probes we have our own in the SOE build image.

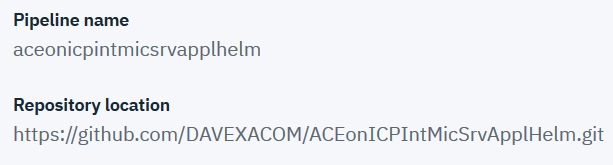


Metrics-service-ace.yaml

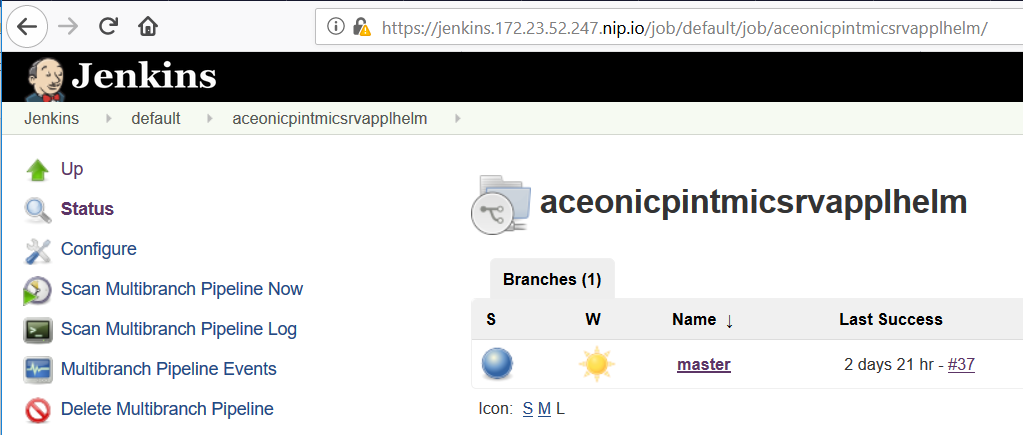


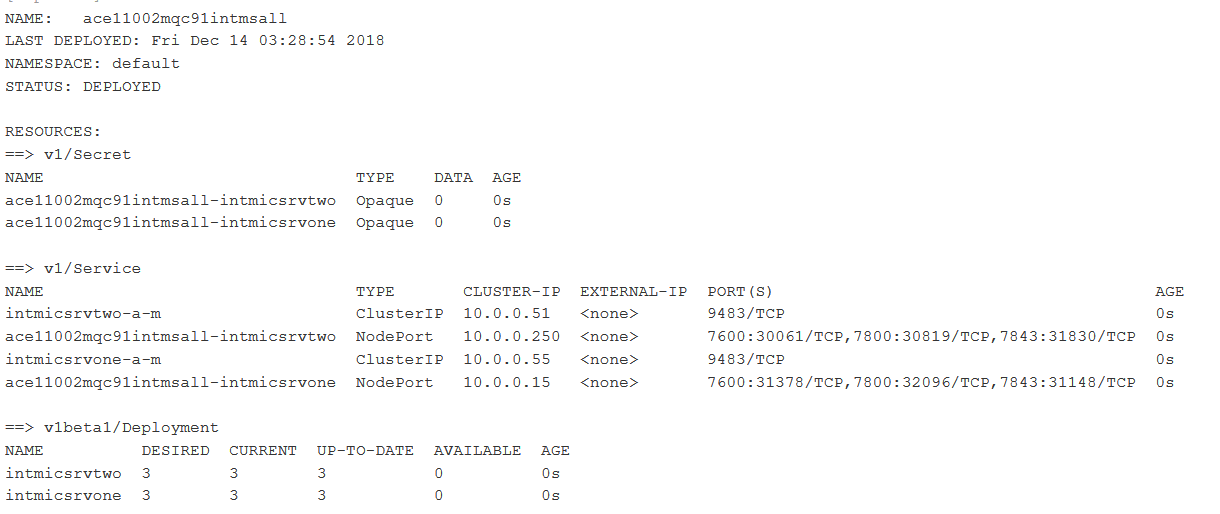
### Microclimate project aceonicpintmicsrvapplhelm



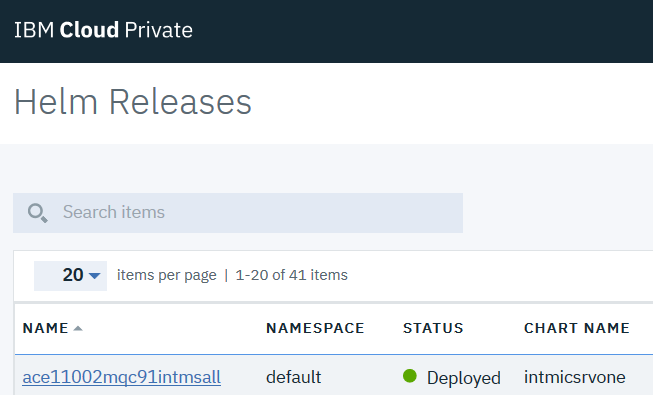


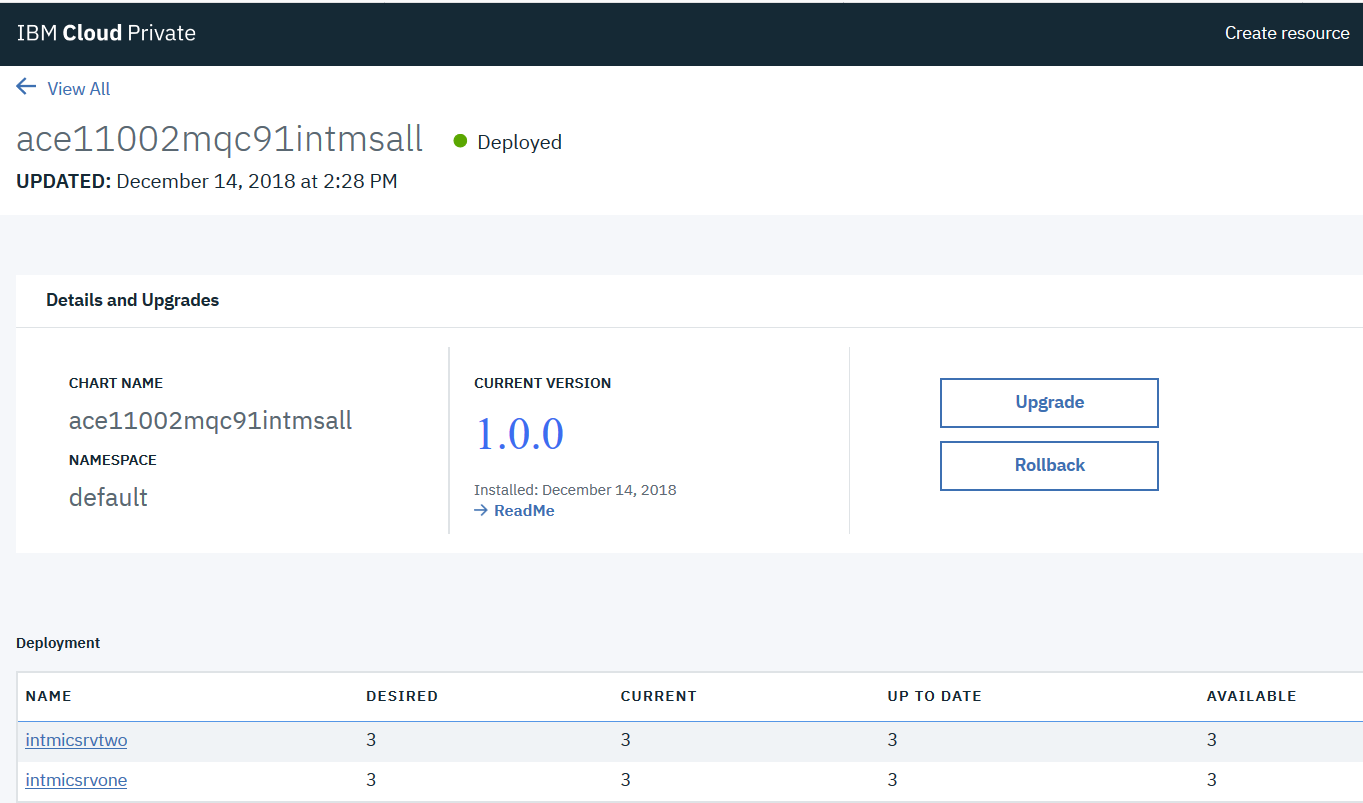
### Jenkins Pipeline - aceonicpintmicsrvapplhelm

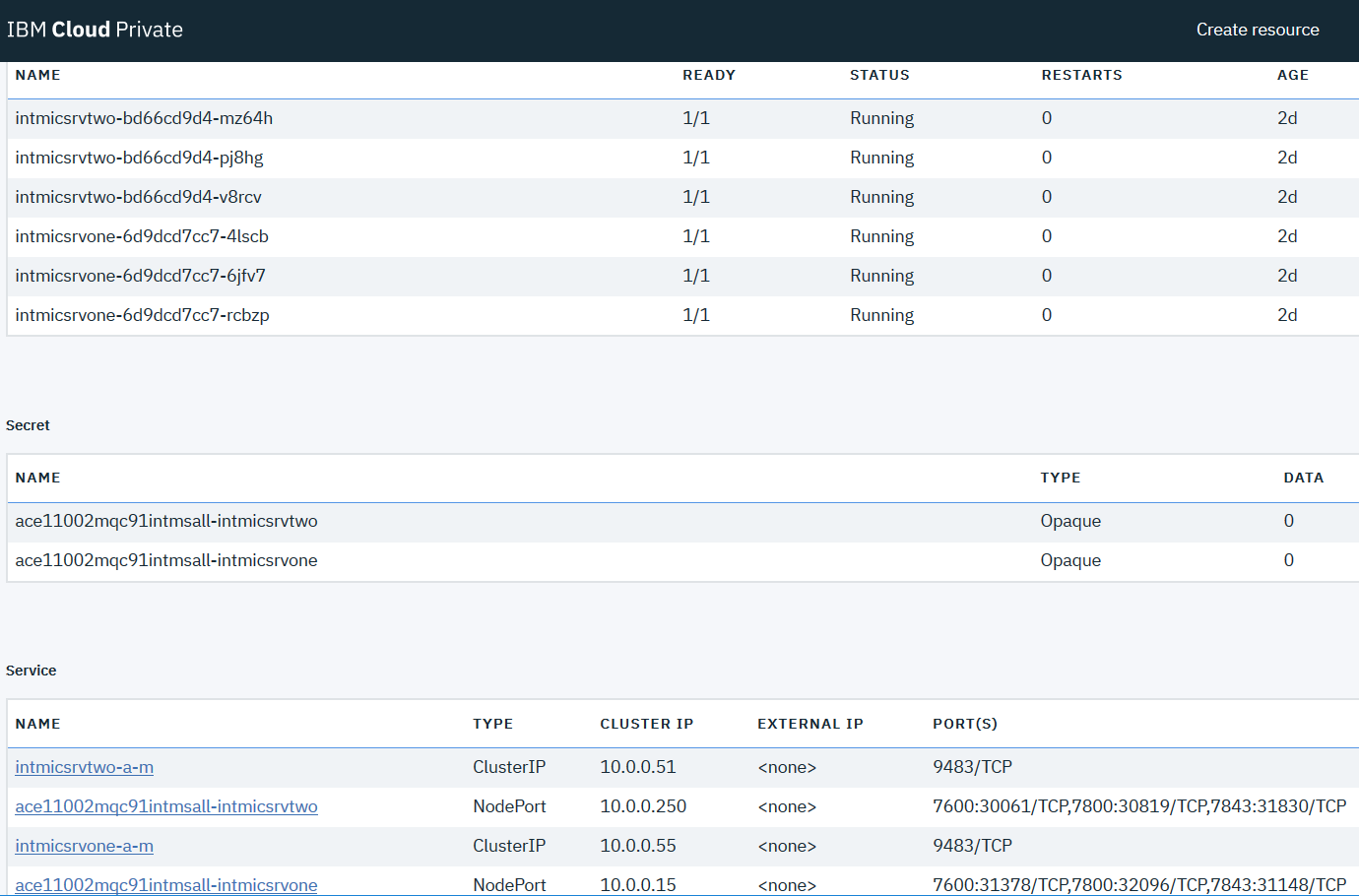




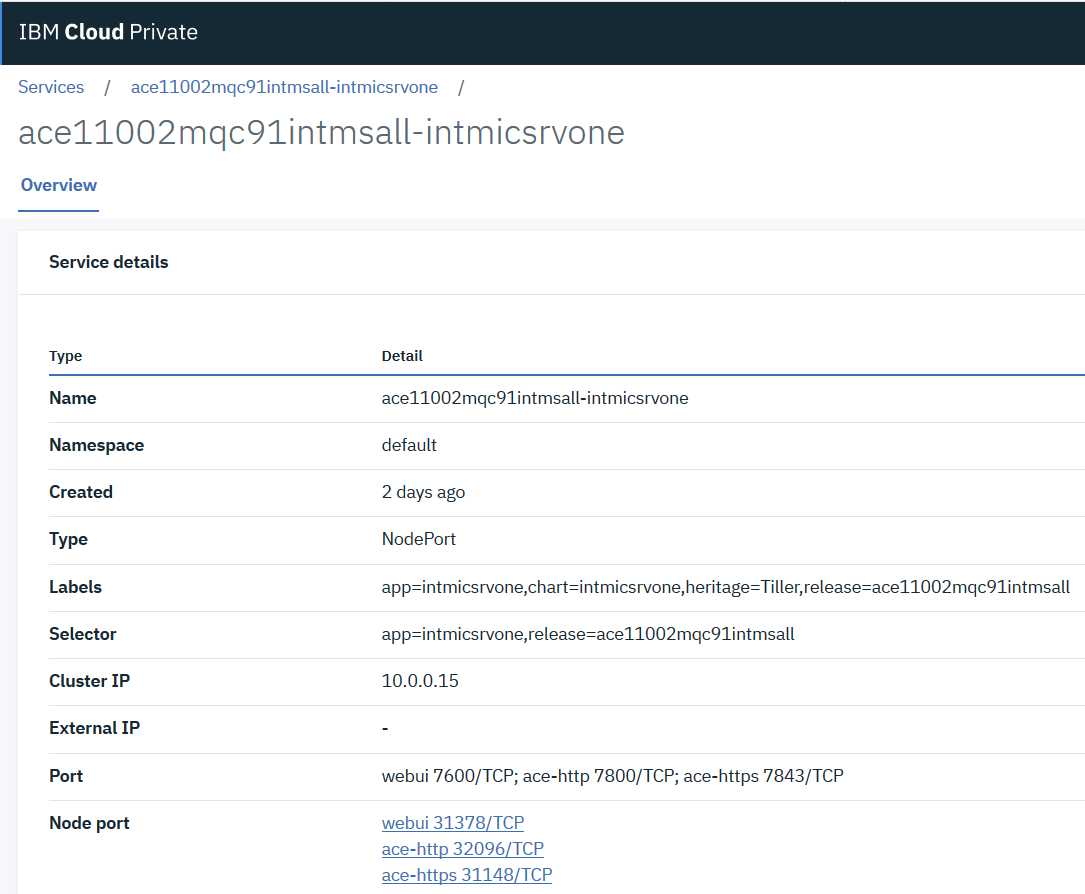
### Helm Release on ICP for Integration Micro Services Application

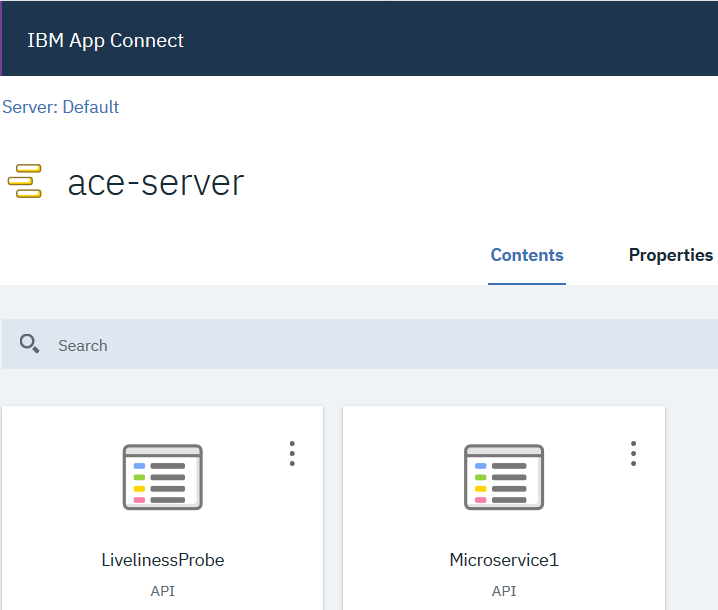






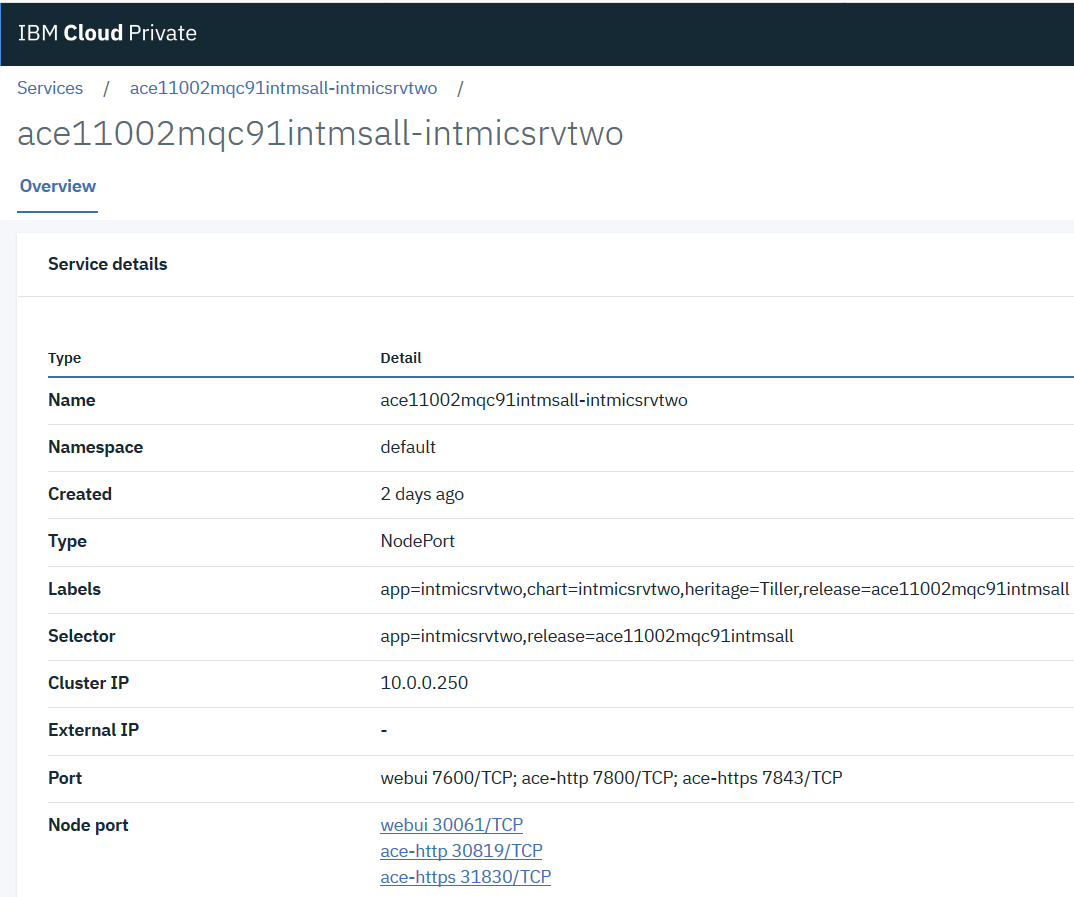
### Integration Micro Service One



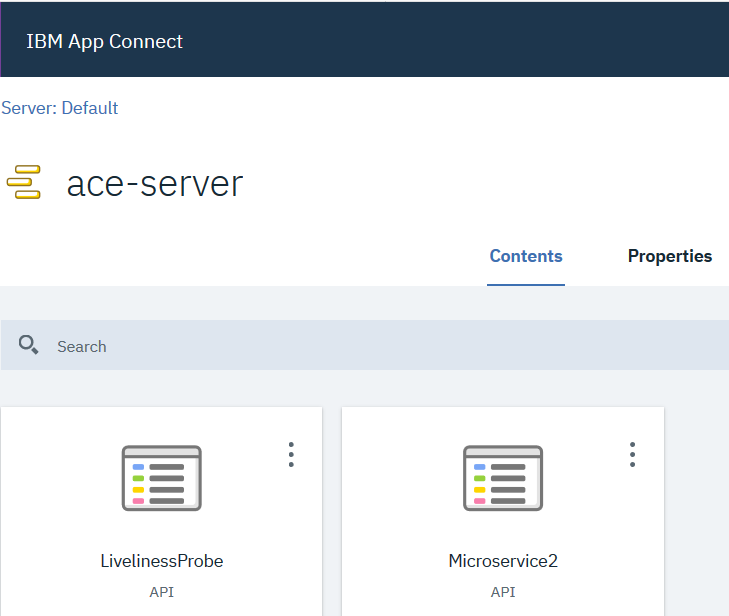


### Integration Micro Service Two

Lets take a look at integration micro service 2



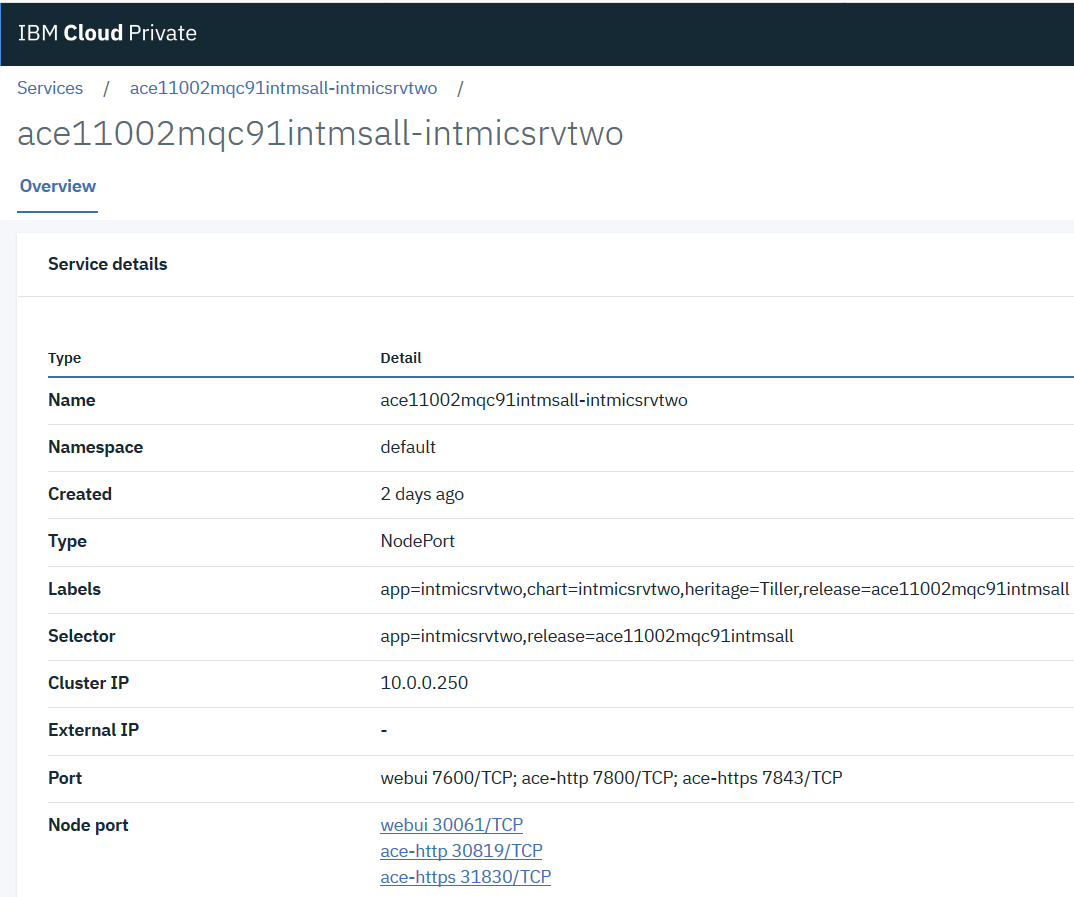
Select the webUI



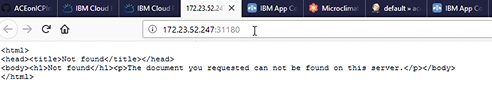
Note we have the LivelinessProbe and MicroService2

### Testing the Integration Micro Services Application

Return to the list of service details for the micro service 2 service.



A quick way to get the URL for the ACE HTTP listener is to click on the link to bring up a browser window from which you can copy the actual URL

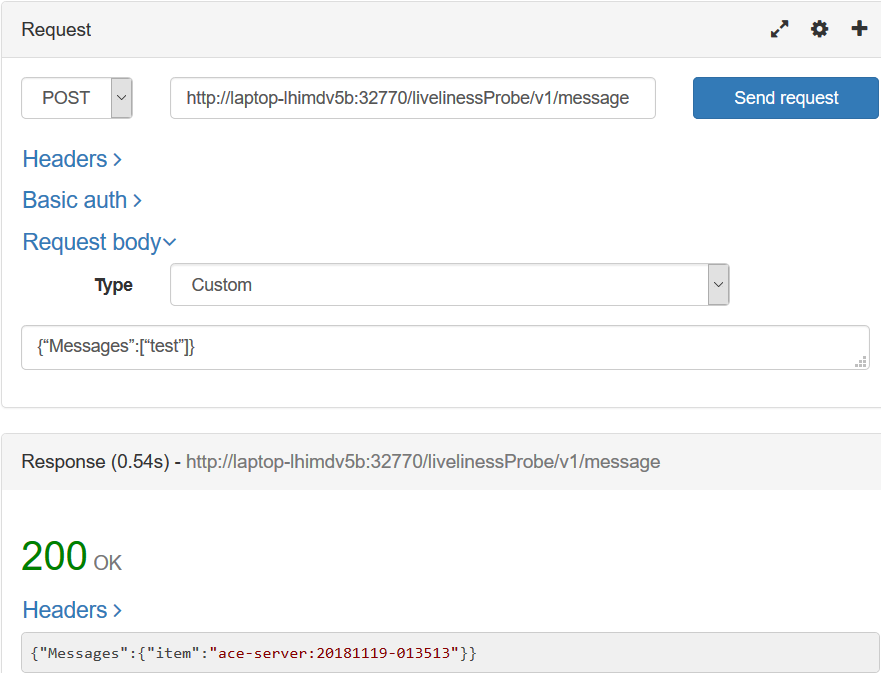


You can use the IPAddress and port number in the URL on the HTTP POST in a REST client.

Then you can test the Liveliness Probe

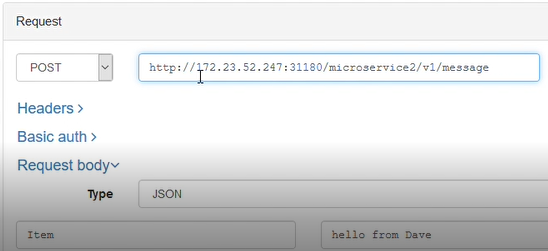
input: {“Messages”:[“test”]}

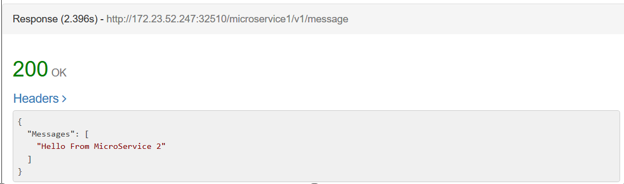
Example URL: http://172.23.52.247:31180/livelinessProbe/v1/message



Note it returns the integration server name plus a current timestamp

And you can test MicroService 2



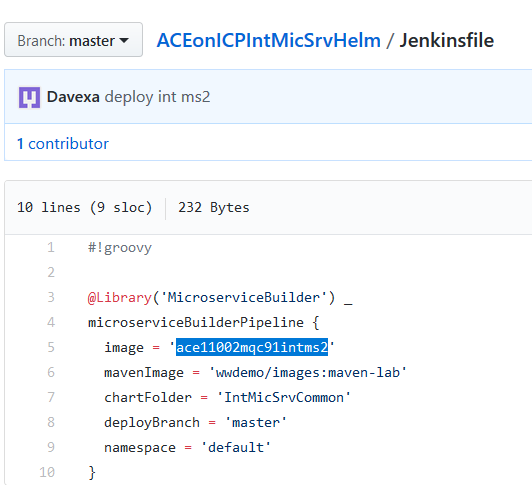


## Deploying Micro Service 2 on it’s own for testing

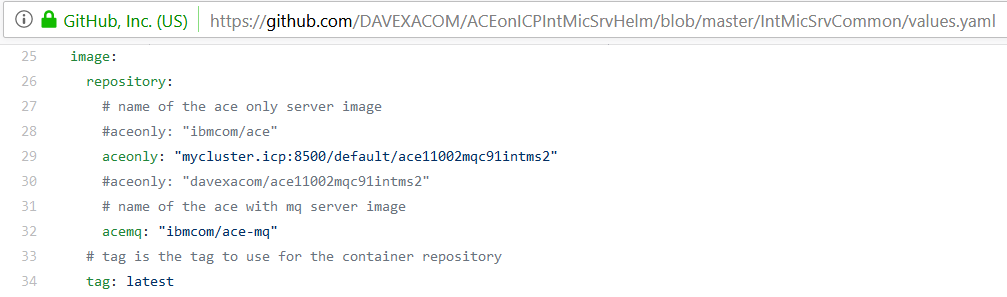
### Source Github repository - ACEonICPIntMicSrvHelm

<https://github.com/DAVEXACOM/ACEonICPIntMicSrvHelm>

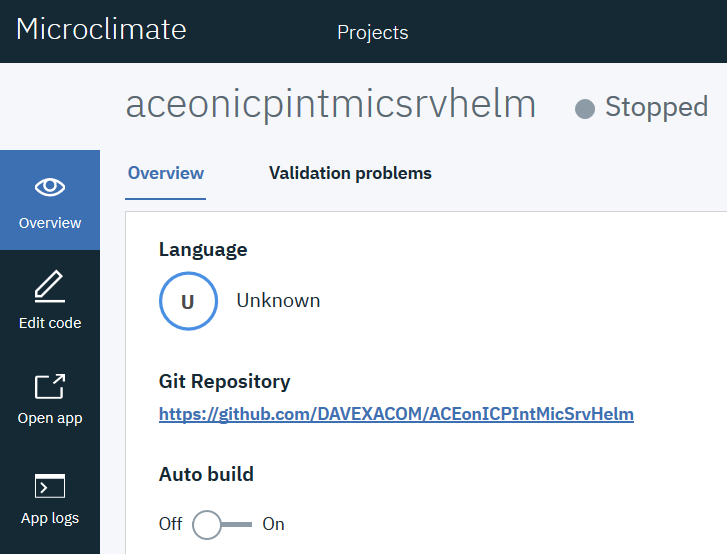
Use this Helm Chart Repos to toggle between Micro Service 1 and Micro Service 2 helm release deployments for testing.

Modify the Jenkins File 

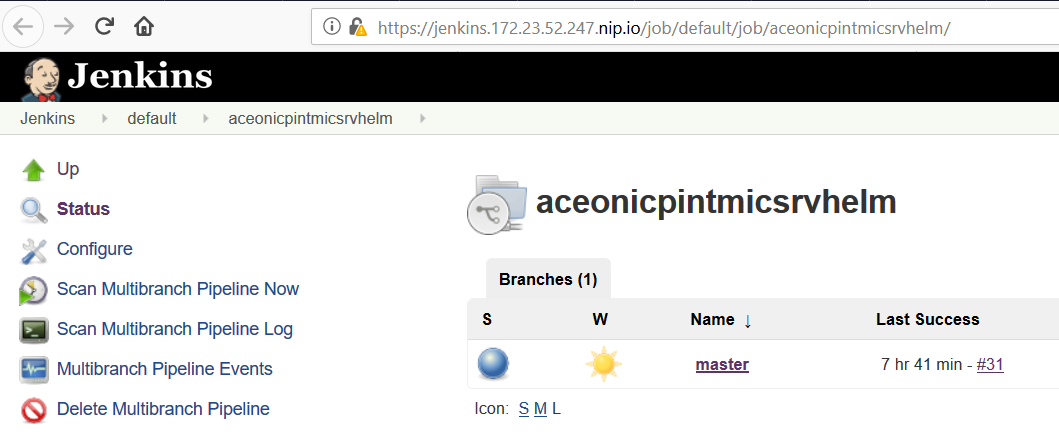
Modify the Values.yaml file

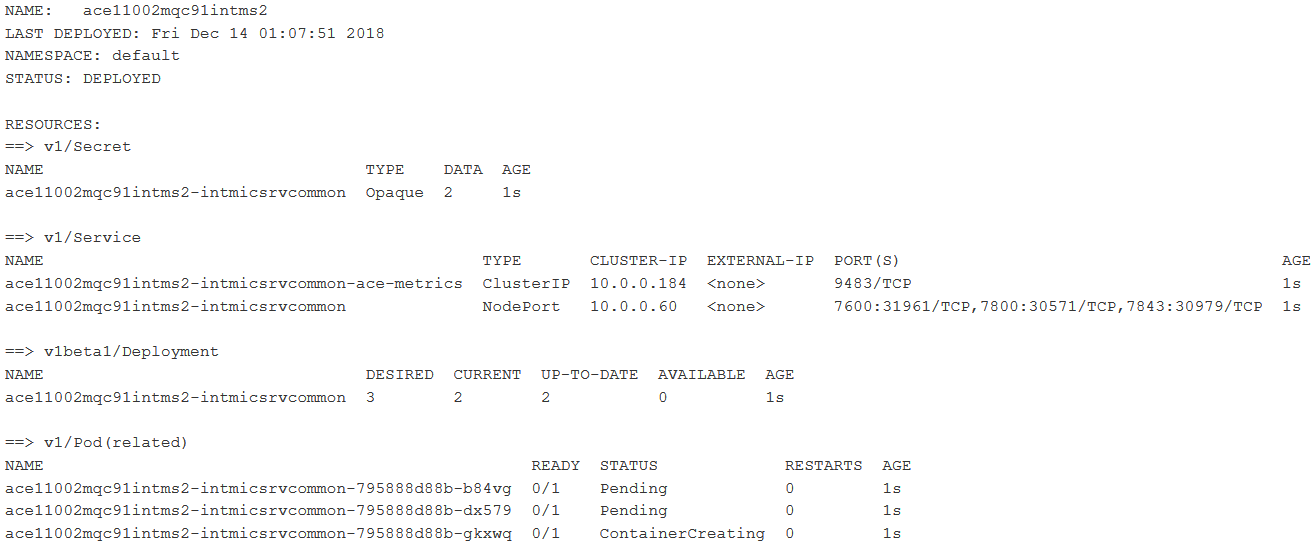


### Microclimate Project



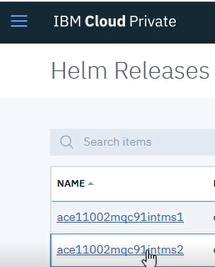
### Jenkins Pipeline



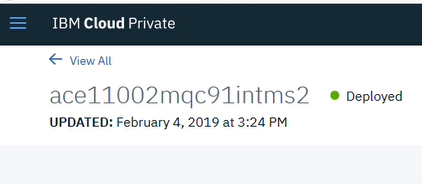


### Testing Integration Micro Service 2 standalone

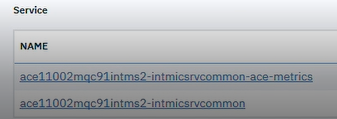
From ICP Console -> Workloads->Helm Releases

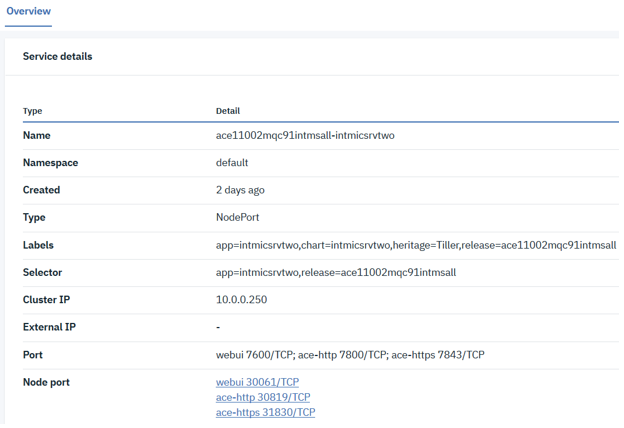


Select the Integration Microservice 2 release

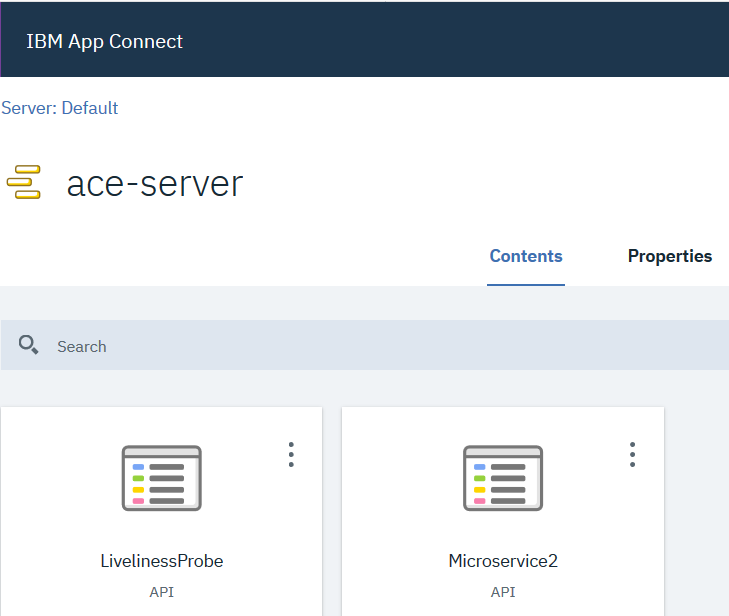


Scroll down to services and select the link (not the ace-metrics link)



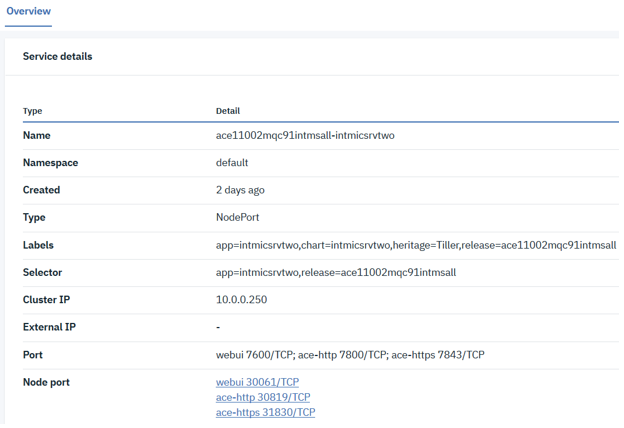


Select the webUI to bring up the ACE WebUI

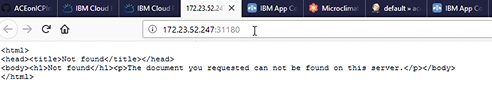


The WebUI verifies that the Liveliness Probe and Microservice2 are deployed

Return to the service details



A quick way to get the URL for the ACE HTTP listener is to click on the link to bring up a browser window from which you can copy the actual URL

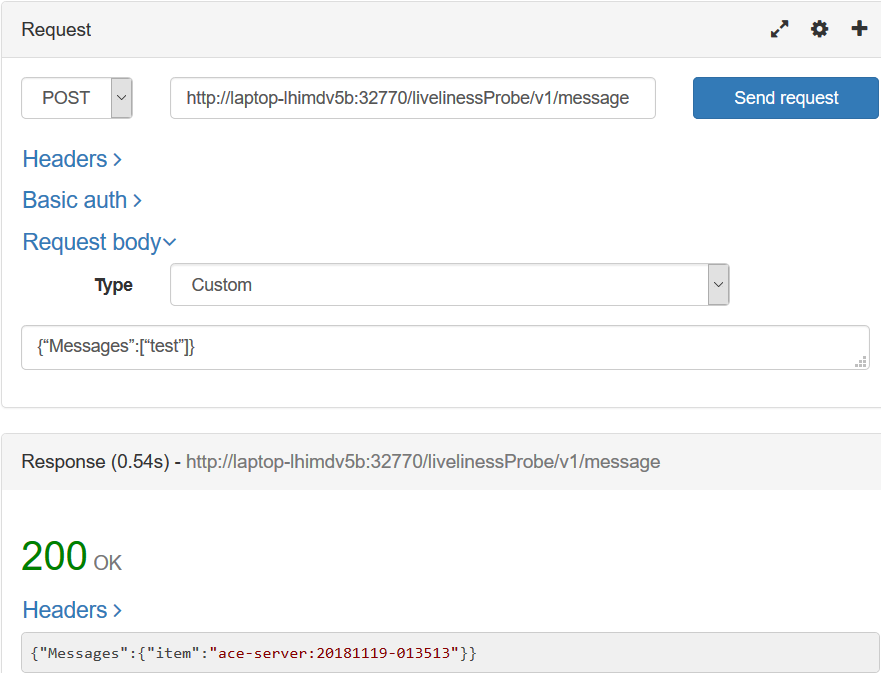


You can use the IPAddress and port number in the URL on the HTTP POST in a REST client.

Then you can test the Liveliness Probe

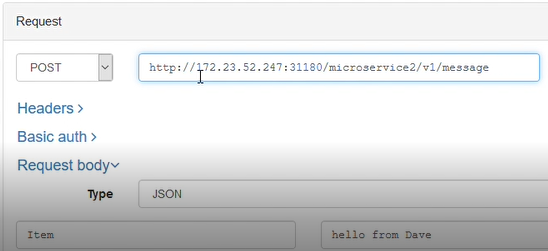
input: {“Messages”:[“test”]}

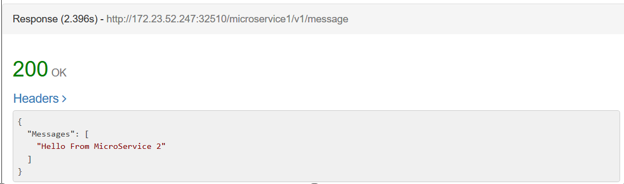
Example URL: http://172.23.52.247:31180/livelinessProbe/v1/message



Note it returns the integration server name plus a current timestamp

And you can test MicroService 2 in a similar way.



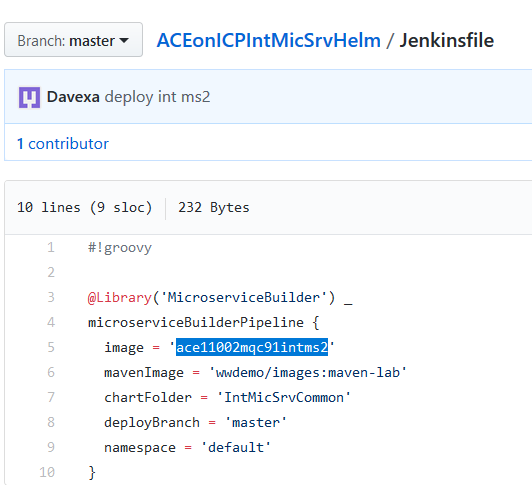


## Deploying Micro Service 1 on it’s own for testing

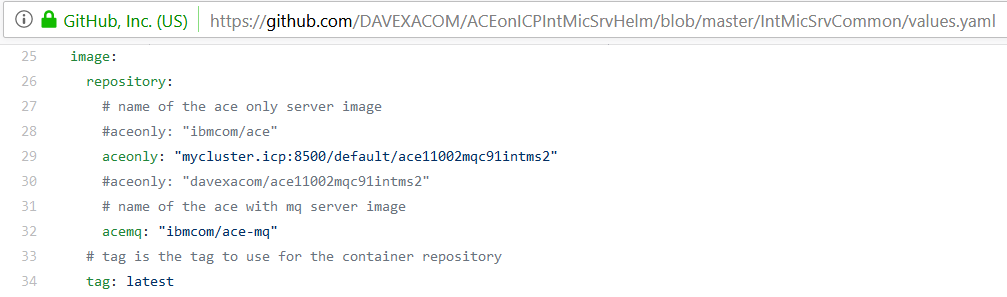
### Source Github repository - ACEonICPIntMicSrvHelm

<https://github.com/DAVEXACOM/ACEonICPIntMicSrvHelm>

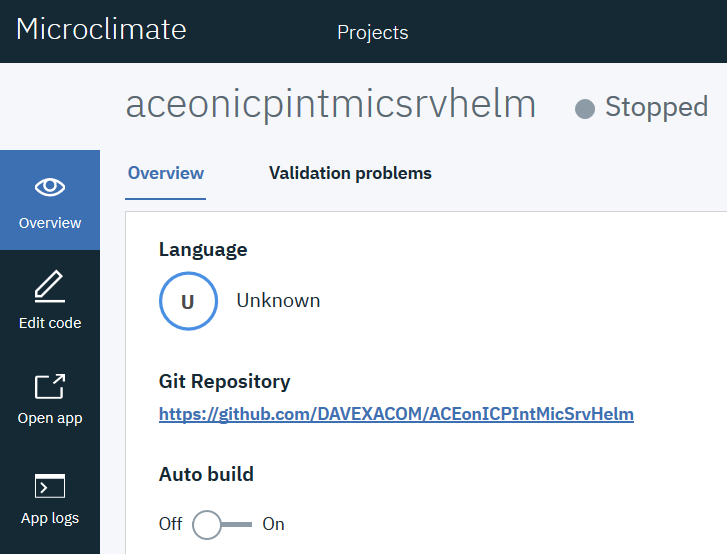
Use this Helm Chart Repos to toggle between Micro Service 1 and Micro Service 2 helm release deployments for testing.

Modify the Jenkins File (in this example you’ll need to switch from 2 to 1)

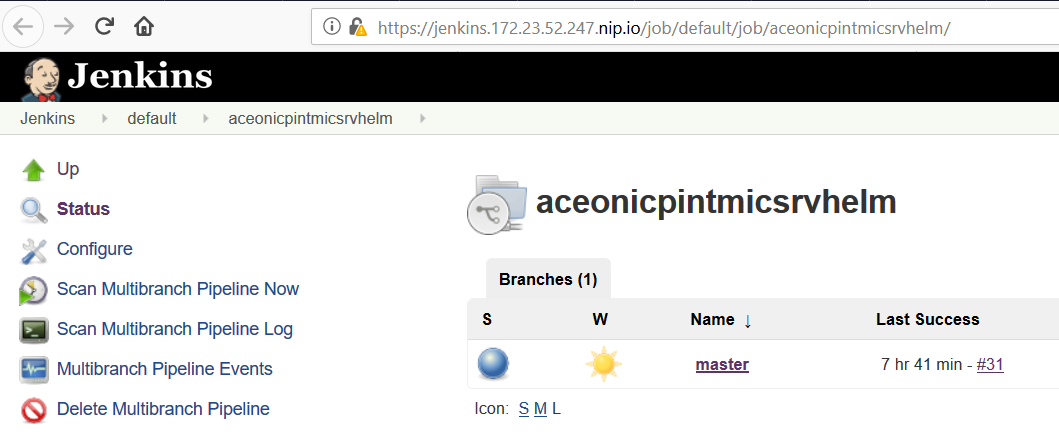
Modify the Values.yaml file

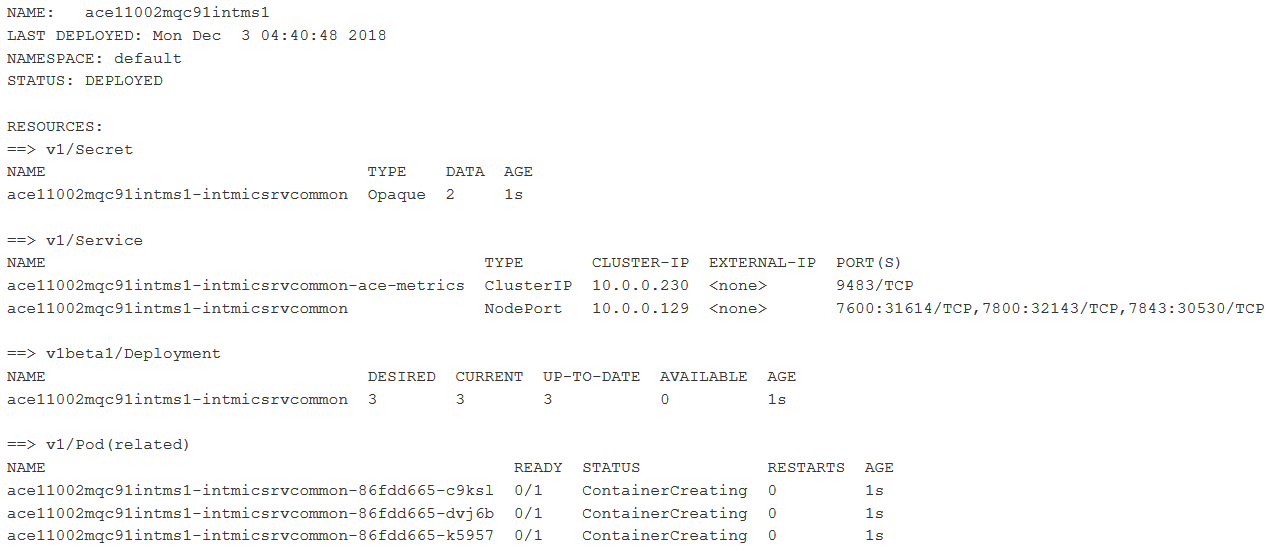


### Microclimate Project

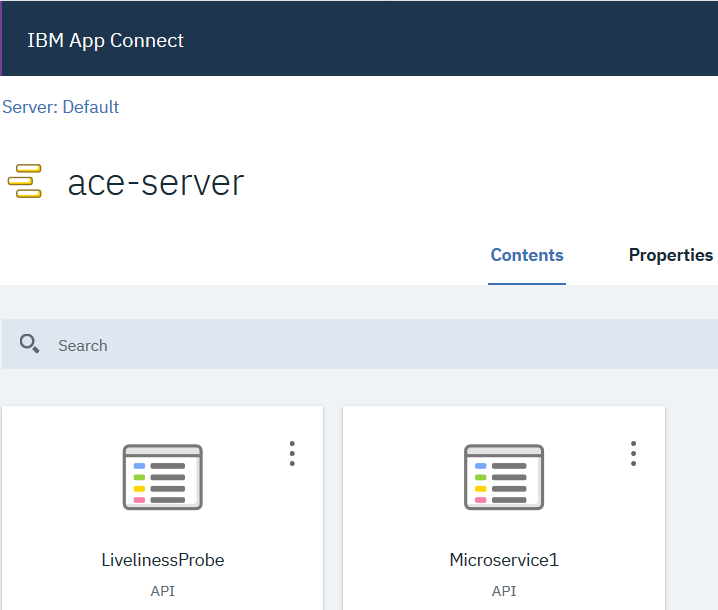


### Jenkins Pipeline





Explore the helm release on ICP and navigate to the services and open the ACE WebUI



### Testing Integration Micro Service 1 standalone

Micro service 1 is designed to call micro service 2 so testing it standalone will fail because micro service 2 is not deployed. You can however test the liveliness probe.

### Testing Liveliness Probe

Note it returns the integration server name plus a current timestamp for input: {“Messages”:[“test”]}

