**Steps for App deployment (MDM)**

Note: -

1. I have considered NodeJS tech stack for the documentation purpose. Similarly, we can do it for others.
2. Also, I’m considering it for “Dev” environment. Similarly, we can do it for “SIT”, “UAT”, “Prod” and “Prod-dr” environments as per the requirement.

Step1: -

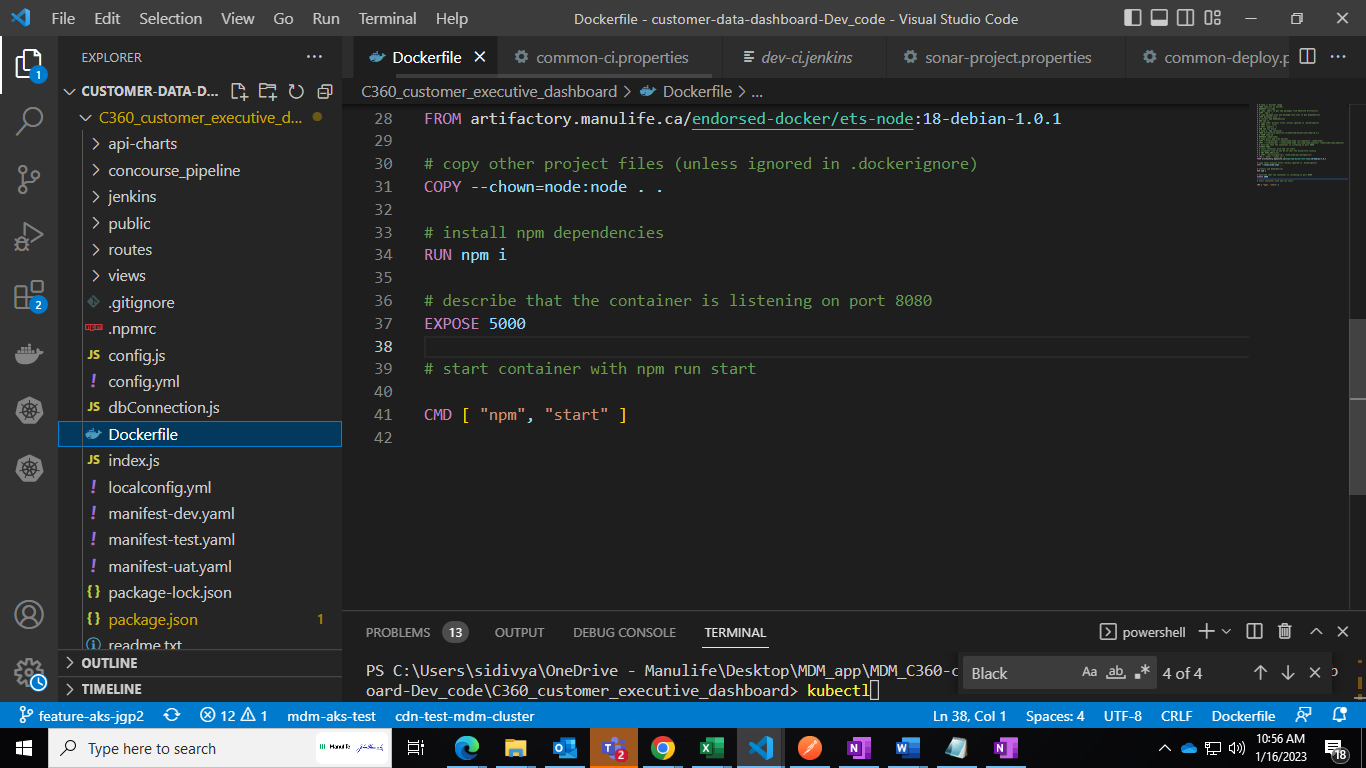
1. Create the branch (e.g. feature-aks-migration for “DEV”) from the base branch (e.g. Develop) branch.
2. Clone the code from GitHub (in any local folder of your choice).
3. Open the code in VS Code Editor (Don’t forget to change the branch to “feature-aks-migration” branch)

Step2: - Code Refactoring

Refactor the code as per the requirement (for PCF to AKS migration).

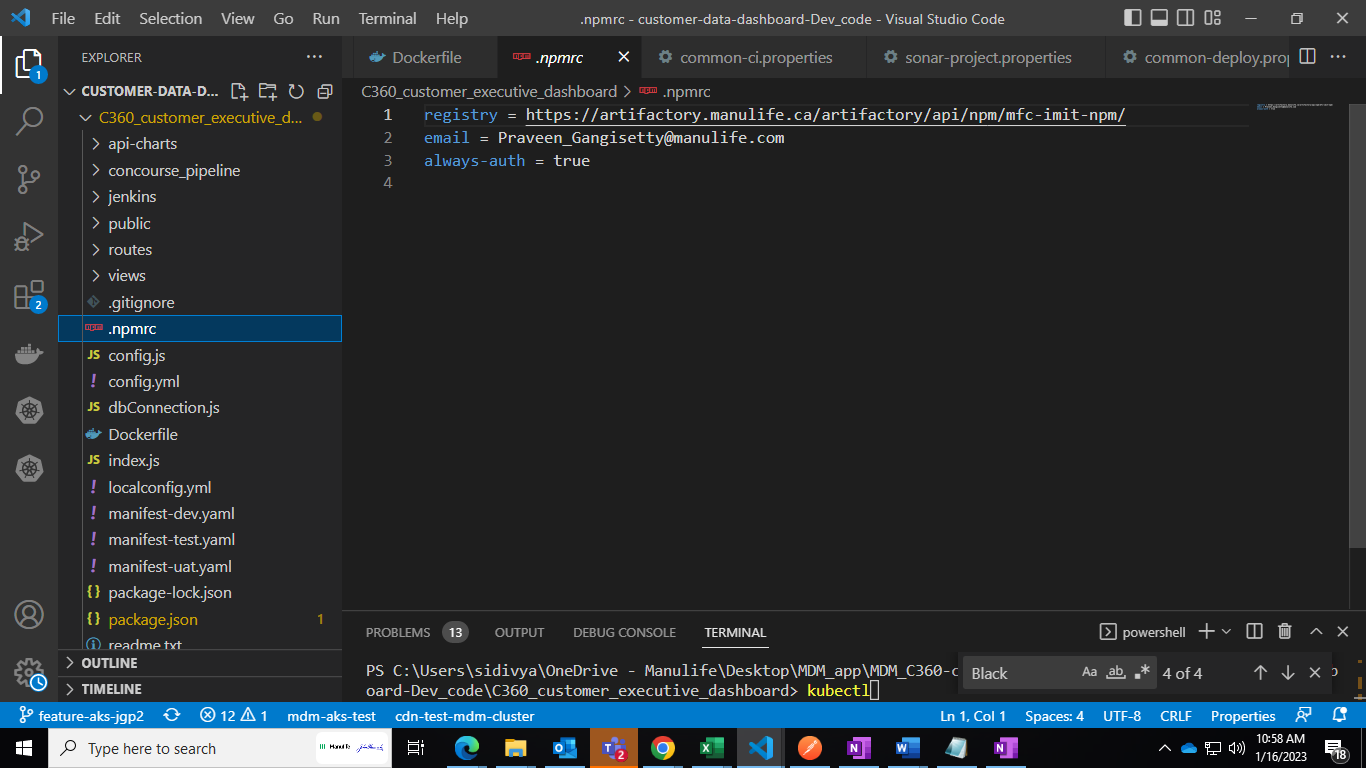
Step3: -

1. Create docker file in the root folder



1. Create .npmrc file in the root folder (put your credentials from artifactory)

Artifactory link:- <https://artifactory.manulife.ca/ui/packages>

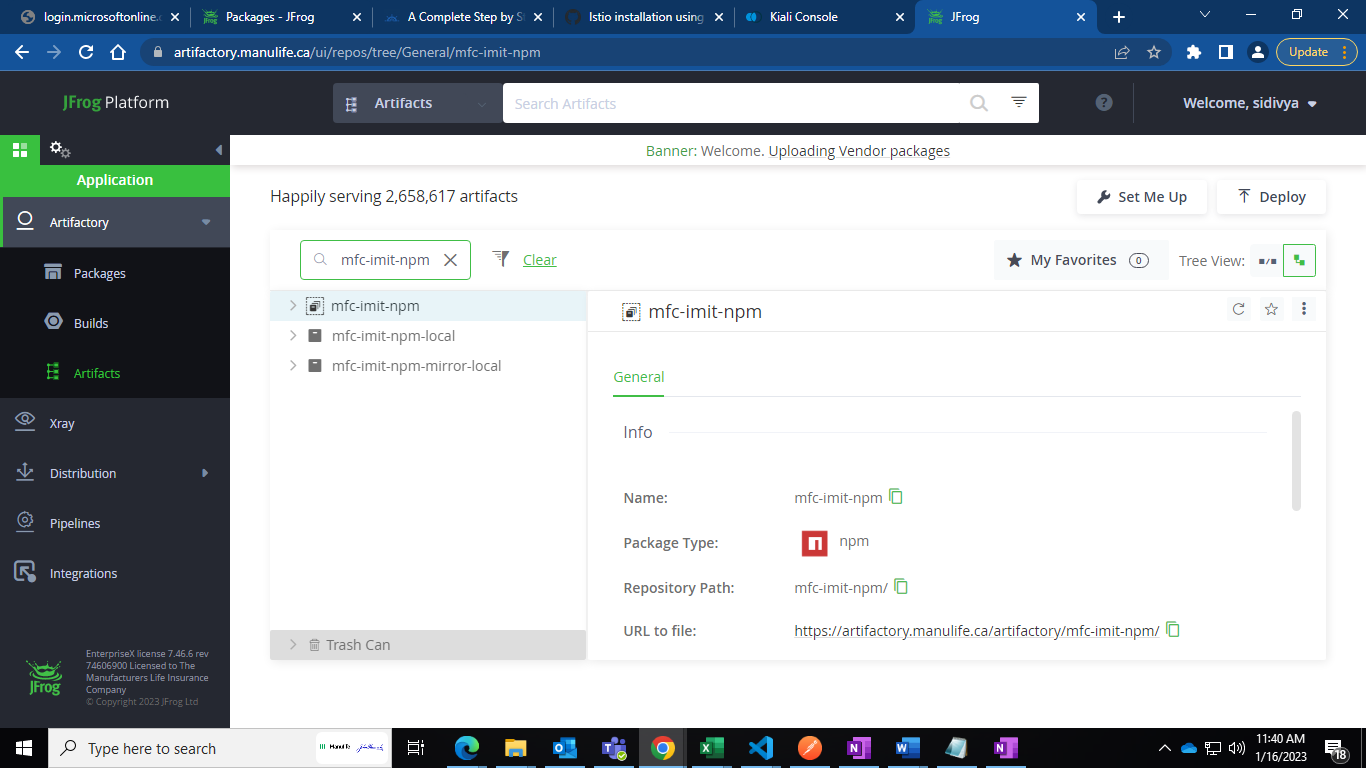


Steps to create “.npmrc” file: -

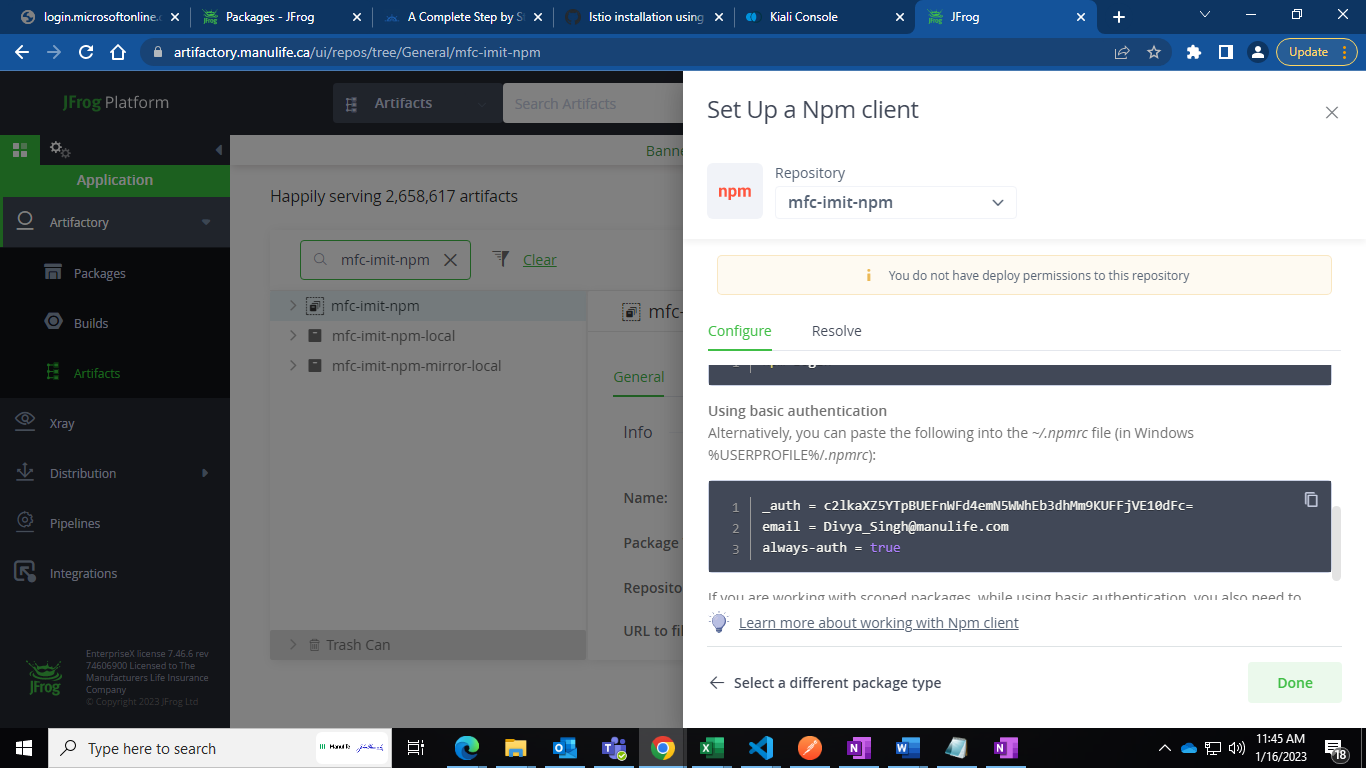
1. For particular Business Unit(BU) find out the “artifact name” e.g. for MDM BU artifact name is “mfc-imit-npm”.
2. Follow above artifactory link and follow

Application🡪 Artifactory🡪 Artifact

now put artifact name e.g. “mfc-imit-npm” in “filter repository”



1. Now click on “Set me up” and enter your password
2. Now pickup “registery“ and “authentication” details and put in .npmrc file in the above format.



Step4: -

Local Dockerization: -

Run the following command in terminal: -

1) docker build -t [any name based on application (e.g. x)] .

2) docker run -it --publish-all [name of the image which we gave (i.e. x)]

Step5: -

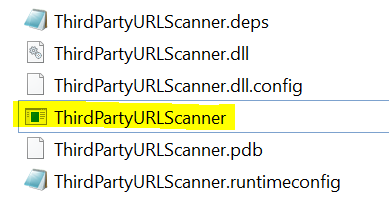
Raising NSG requests for all external URLs which app is trying to connect using the tool as follows:

[Enterprise Technology & Services PMO - 3rd\_party\_URL\_Scanner - All Documents (sharepoint.com)](https://mfc.sharepoint.com/sites/ETSPMO/Active%20Projects/Forms/AllItems.aspx?RootFolder=%2Fsites%2FETSPMO%2FActive%20Projects%2FRetire%20PCF%2FTCS%2FCDN%20Pilot%2F3rd%5Fparty%5FURL%5FScanner&FolderCTID=0x012000C267C37E39CA3D4D8EB82A877B73B2F0)

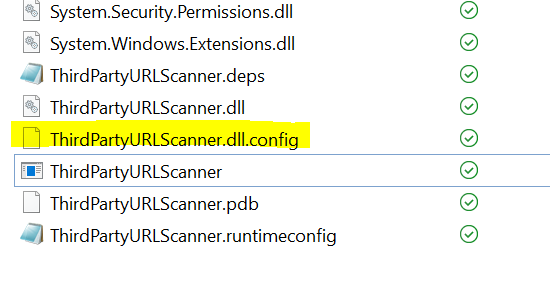
This will help us identify all the integration URLs in the code and we need not manually check each configuration files or code file.

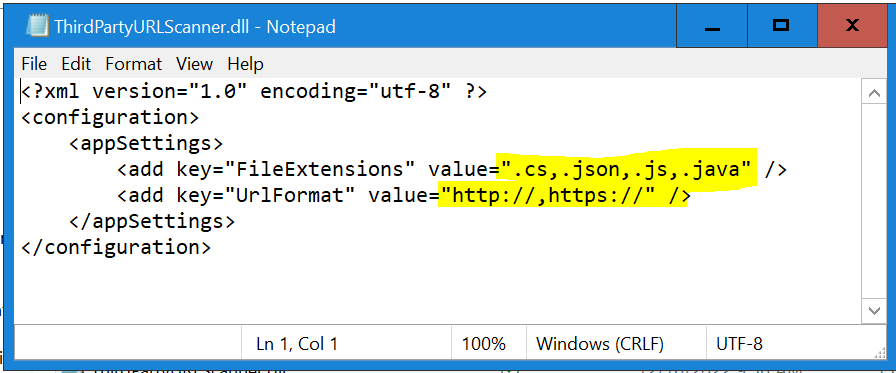
Below are the steps to run the Tool

1. Unzip the Folder
2. Double click on the below file



1. Once you run the exe you will be prompted to provide the Path of the Local Project Folder to Scan
2. Please provide the Root folder path of your respective Project folder
3. Once the execution is completed there will be “URL.txt” file created at the root folder where all the URLs in the code will be extracted from the Code files.
4. You can also modify the file extensions which your project can have for eg. “.cs,.java,.yaml,yml”. I have already configured this but you can also add new file extensions based on your Tech Stack.
5. You can also add or modify the protocols that needs to be added in scan for eg. “http,https,amqp”. I have already added this in the configuration file.





*Note: Please install .net6.0 in case this tool does not work on your respective machine.*

**After getting all the URLs, perform “nslookup” command in cmd to get the Ips for the hosts.**

**Now reach out to respective DevOps team for raising the requests.**

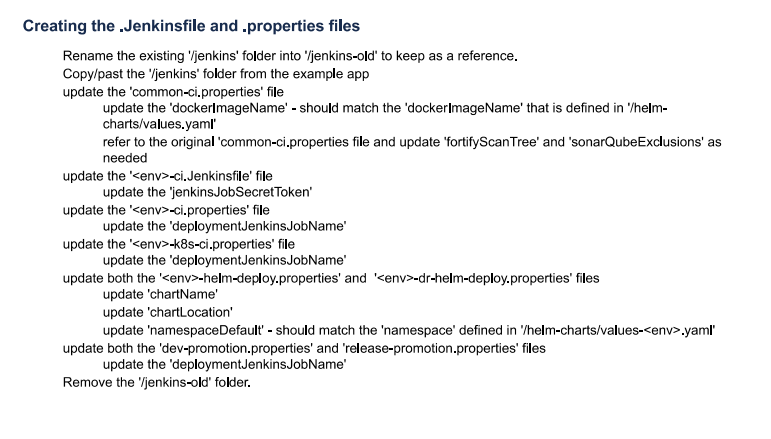
Step6: -

Adding values in AKVs

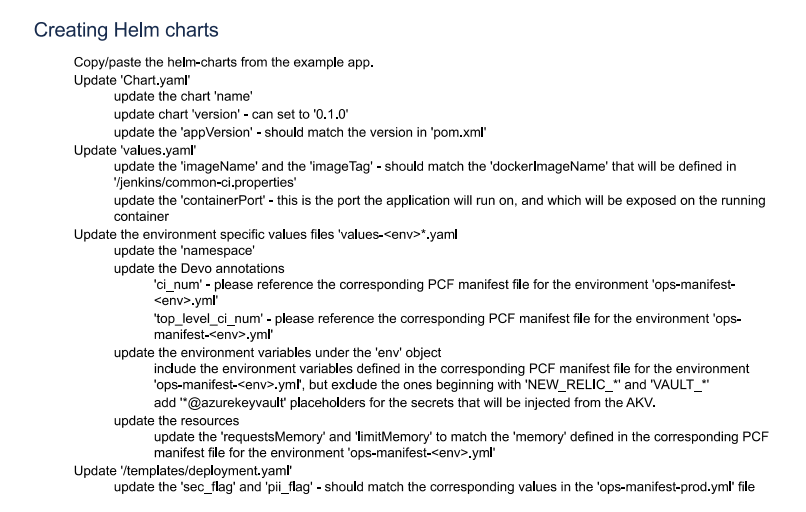
1. We need to get cluster Infra details from DevOps team (eg. AKVs link, namespace, cluster name etc.)
2. We must put sensitive variable names in “akvs” in “values-dev.yml” and need to insert values of these variables in AKVs.
3. Non-sensitive details can be kept under “configmap” in “values-dev.yml”

Step7: -

1. Create Jenkins folder in the root directory of the project.

* Please check all the files of “Jenkins” folder of existing project and modify the details as per your cluster details and “Pipeline” names.
* 

1. Create “api-charts” folder in the root directory of the project.

* Create this folder from existing project and modify all the files as per the project details (e.g. values-dev.yml)
* 

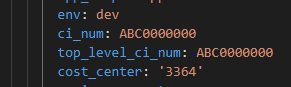
1. Create “sonar-project.properties” file in root directory of the project.

* Copy this file from the existing project from respective tech stack project.

Step8: -

Devo integration: -

* For Devo integration we need to get “CI\_NUM” and “TOP\_LEVEL\_CI\_NUM” from customer and update in “value-dev.yml” file.



NewRelic integration: -

* We have to add “newrelic” package in “package.json”
* Then we have to import “newrelic” package in “index.js”
* Now we need to add “newRelicAppName” and NEW\_RELIC\_LICENSE\_KEY under “configmap” and “akvs” respectively in “values-dev.yml”

Step9: -

Push the code to repository.

Use below commands to push the code to respective branch

* git add -A
* git status
* git commit -m [“appropriate message”]
* git pull
* git push

Step10: -

Jenkins Pipeline creation:  
Key Points:

**A** - Three pipelines need to be created for Jenkins Jgp 2.0. Reference has been taken from one of the MDM Application  **STANDARDIZE\_PERSONNAME.** The information is given below:

1. **IMIT\_C360\_STANDARDIZE\_PERSONNAME\_DEV\_CI**
2. [**IMIT\_C360\_STANDARDIZE\_**PERSONNAME**\_K8S\_DEV\_CI**](https://jenkins.manulife.com/job/IMIT_Projects/job/IMIT_API_TEAM/job/CDE_C360_API_AKS/job/IMIT_C360_STANDARDIZE_PERSONNAME_K8S_DEV_CI/)
3. **IMIT\_C360\_STANDARDIZE\_PERSONNAME\_K8S\_HELM\_DEV\_CD**

**B** - These are some Naming Standards we need to follow   whenever creating the pipelines under Jenkins JGP 2.0:

* **<BU>\_<Application Name>\_<ENV>\_CI**
* **<BU>\_<Application Name>\_K8S\_<ENV>\_CI**
* **<BU>\_<Application Name>\_< K8S\_HELM\_<ENV>\_CD**

In place of <ENV> place holder the below values are only acceptable:

**\_DEV\_|\_DEVELOP\_|\_UAT\_|\_PROD\_|\_PRD\_|\_TST\_|\_TEST\_|\_RELEASE\_**

* **Points Needs to be remembered while creating Jenkins jgp 2.0 jobs:**

1. Do not create any job using SIT naming convention as it will not take other naming standards and pipeline will fail.

1. The pipelines **naming standard** should start with the folder name followed by Application name and then the naming standard as suggested above.    
      
   NOTE: Please do not change anything from the naming standards or else it will cause pipeline failure.

1. Please create as Separate branch for Jgp 2.0 to make sure that we can segregate jgp 1.0 and 2.0. For the reference, please check the Repo link given below:

[manulife-ca/standardizepersonname at feature-aks-jgp2 (github.com)](https://github.com/manulife-ca/standardizepersonname/tree/feature-aks-jgp2)

1. Please ensure that for node.js tech stack we are adding the Sonar Properties file or else it will not allow us to run the first pipeline DEV\_CI and will throw the error.
2. Ensure that in the **Values.yaml** we are putting docker tag value in Single quote for **Ex**-  apiDockerTag: '{{<Application name>}}'

**Reference Link for Node.Js Tech Stack:**

[All [IMIT » IMIT\_API » CDE\_C360\_API\_AKS » IMIT\_MDM\_C360\_StandardizePhoneNumber\_NonProd] [Jenkins] (manulife.com)](https://jenkins.manulife.com/job/IMIT_Projects/job/IMIT_API_TEAM/job/CDE_C360_API_AKS/job/IMIT_MDM_C360_StandardizePhoneNumber_NonProd/)

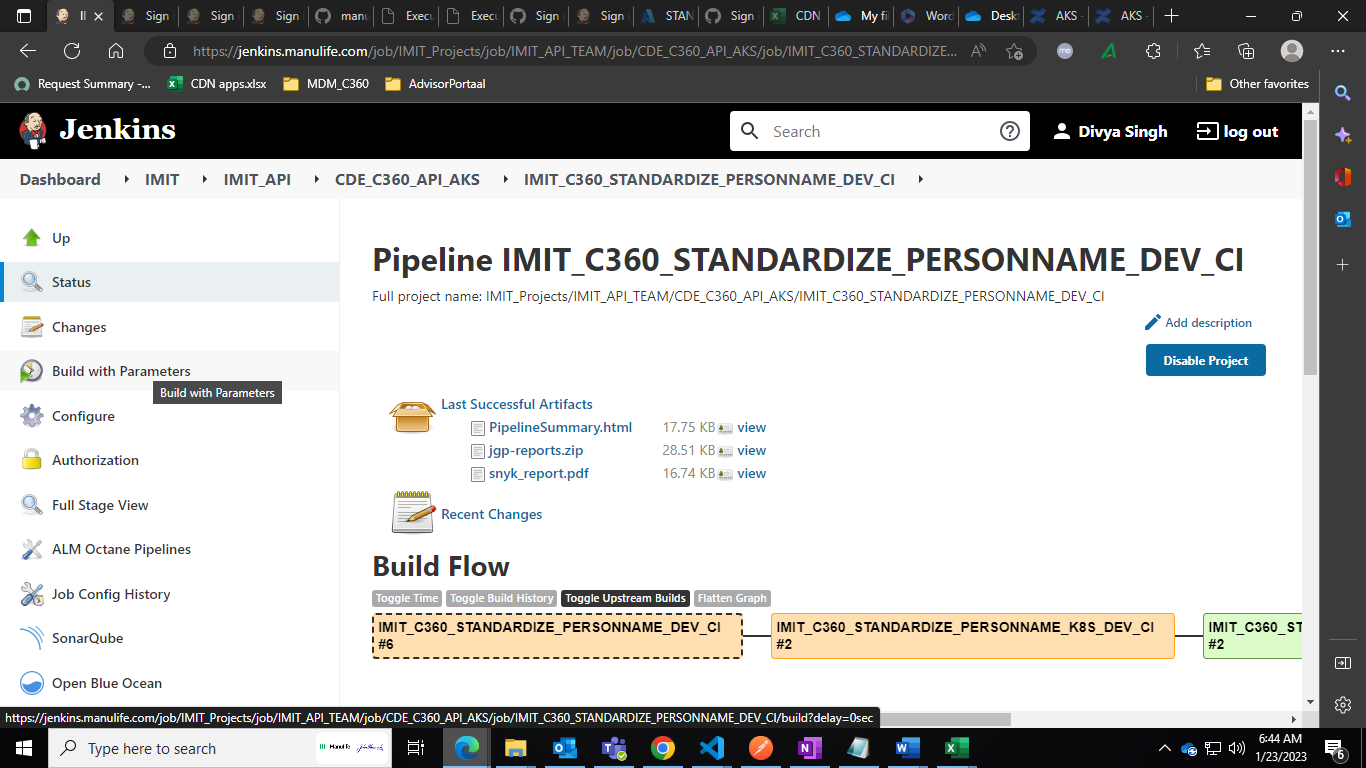
[AKS - Getting Started Guide - CET - Engineering Services - Confluence Canada](https://confluence.manulife.ca/pages/viewpage.action?spaceKey=CETES&title=AKS+-+Getting+Started+Guide)

Step11: -

Running the pipeline

1. Select the 1st pipeline i.e.( **IMIT\_C360\_STANDARDIZE\_PERSONNAME\_DEV\_CI**

) and click on “Build with Parameters”.



1. Remaining pipeline will run automatically if there is no issue in configuration.

Step12: -

Checking the pods logs :-  
 1. After running all the pipelines successfully, we need to check the “pod” logs. Run the following command to check the log: -

* Kubectl get pod
* Kubectl logs [your application pod name] -f

Step13: -

Checking istio-proxy logs

* Kubectl logs [your application pod name] -c=istio-proxy

Check the blackhole clusters in logs and collect the IPs and Ports.

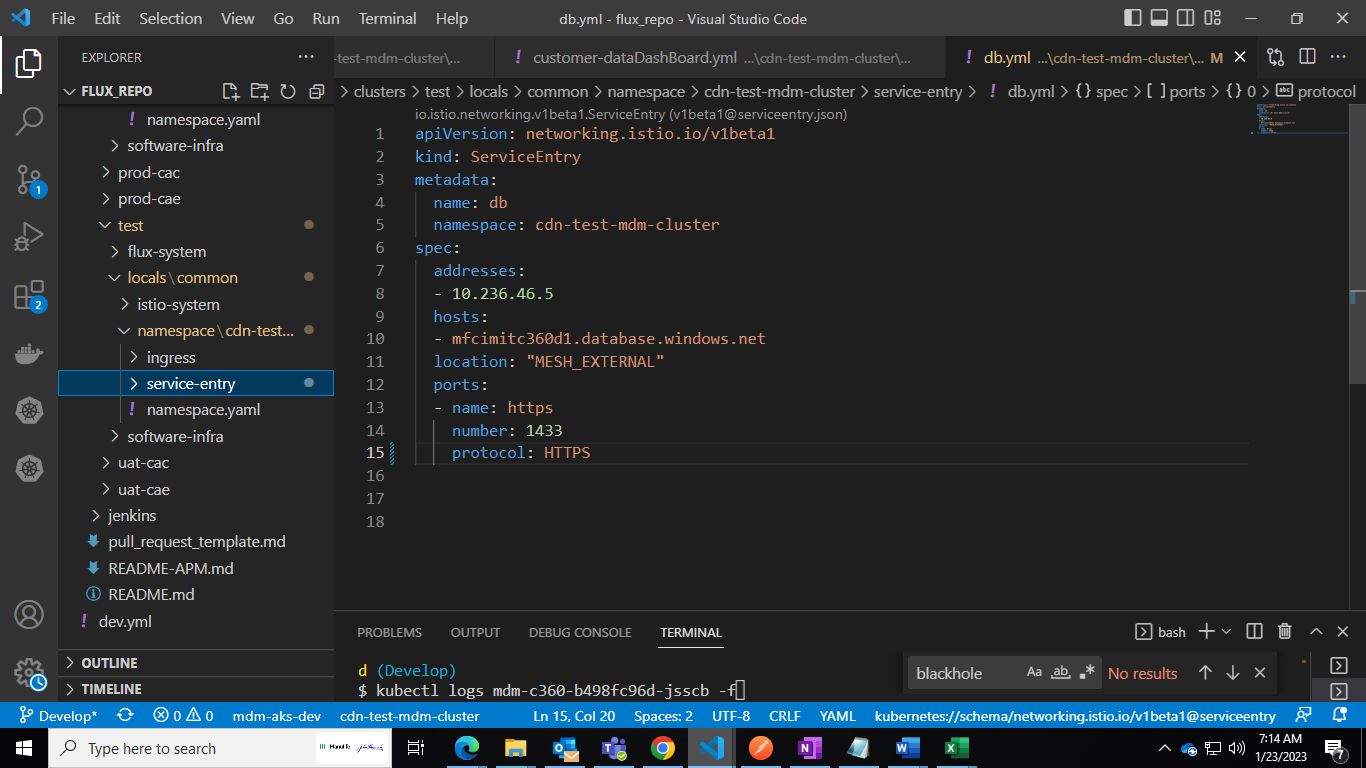
Step14: -

Adding IPs to flux repo

After collecting IPs from above steps, we need to add those IPs in flux repo.

* We will get flux repo GitHub link from DevOps team.
* dev\locals\common\namespace\cdn-test-mdm-cluster\service-entry

in this folder create a new file as per your host name and and the below content after modifying as per your requirements.



* Now push the code to Git repo.
* Run “kubectl get ks -Aw” to check the status and wait until all turn to “true”.
* Now repeat “Step13” until all “blackholes” are removed from “istio-proxy-logs”

Step15: -

Using postman collection to test the application: -

* If your app is running in pod and also there is no issue in pod-logs then we need to test our application using “postman collection” which we will get from client.
* For all the endpoints we should get “200” status.