Vajra Onboarding User Guide

Introduction to Vajra

Vajra is a platform which provides an on-demand sandbox environment for application owner/teams to do their functional testing of components as well as Integration testing/E-E system testing.It also helps developer to quickly check the impact of their latest code changes in isolated/sandBox environment as part of CICD flow .

Phases In vajra

Every system which will get deployed in sandbox environment of K8 through vajra platform has to pass through below phases .

On Board

During on-boarding process app owner has to provide all the system related info to the vajra platform in order to create docker image out of it .

Mocking/Stub

If you don't want to onboard the system or whitelist it then we have the option of stubbing the component. This is optional but if you want to include any mocking please create it before design phase .So that it can be included in the pipeline design before pipeline build & deployment .

Design Pipeline

In this phase user can design the replica of their system architecture by connecting all the required onboarded components. It depends on the On-boarding phase as user has to complete all the onboarding of required components for designing a pipeline. After designing user has to SAVE the pipeline for next phase(Pipeline Build) .

Pipeline Build

Once the Pipeline is SAVED in the design phase . We need to trigger a build for the saved pipeline . Once the pipeline build is ready , user can deploy that build .

Deploy

In this phase user deploy the certified build in vajra sandbox environment i.e Kubernetes(K8) .

Testing

If deployment is SUCCESS or all the pods of deployed pipeline comes up . User/ Team can do functional / integration testing as part of CICD flow .

Onboarding the System

In vajra - onboarding means application owner will come up with their system related information and then through Vajra they can create docker images of all the system involved in the architecture automatically.

Why System Onboarding

Vajra deploys the designed pipeline / replica of system architecture in Kubernetes and we know Kubernetes understands only container. So Vajra also uses Docker as container runtime environment for K8 and allows users to containerise their applications. So basically onboarding means containerise your apps for deploying in kubernetes namespace by providing system related information to vajra platform .

Vajra URL For Onboarding

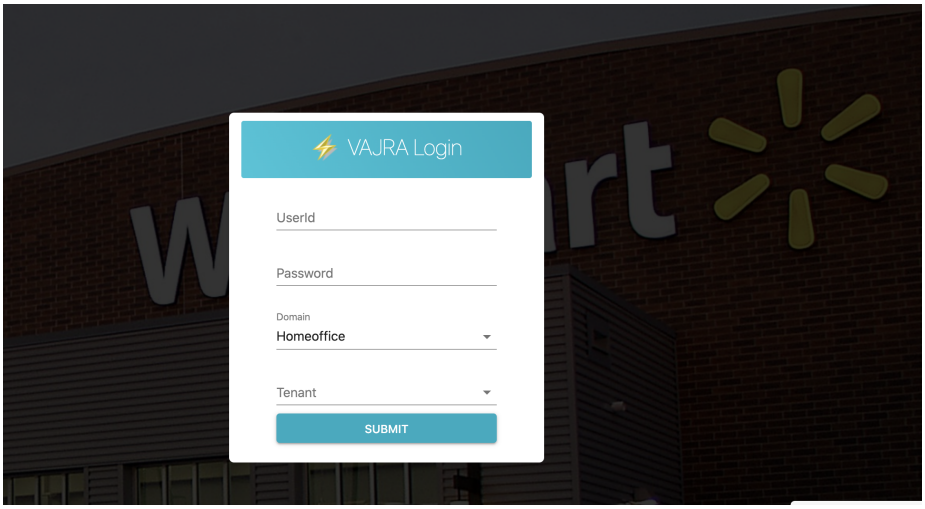
To onboard the system in vajra please use below url :-

https://vajra.walmart.com

Login Restriction

For login into vajra , users needs to contact **#vajra-onboarding** slack channel for creating account into vajra & other login related queries. Vajra also provide role based access for component Onboarding , Pipeline designing & deployment . Currently we support Admin, Design & Deploy user access based on roles.

After user login account created in vajra . You can provide below information in login screen to get access . UserName & Password is your's Walmart credential , Vajra don't store user password as it uses Walmart's IAM API to authenticate directly .



Tech Stack Supported

- TOMCAT & SPRINGBOOT as Services .
- KAFKA & CASSANDRA as DataSource.
- STORM as Platform.
- MONGODB, COUCHBASE , SOLR , DEFAULT (Any Docker Image) as Docker.
- MOCKING through Vajra Stub server which internally uses wiremock server API along with vajra stub APIs.

Ways Of Onboarding

There are 2 options through which user can onboard systems into vajra platform.

1. Onboarding through one Ops Configuration - Supported only for TOMCAT & SPRINGBOOT application .
2. Onboarding through Manual Configuration .

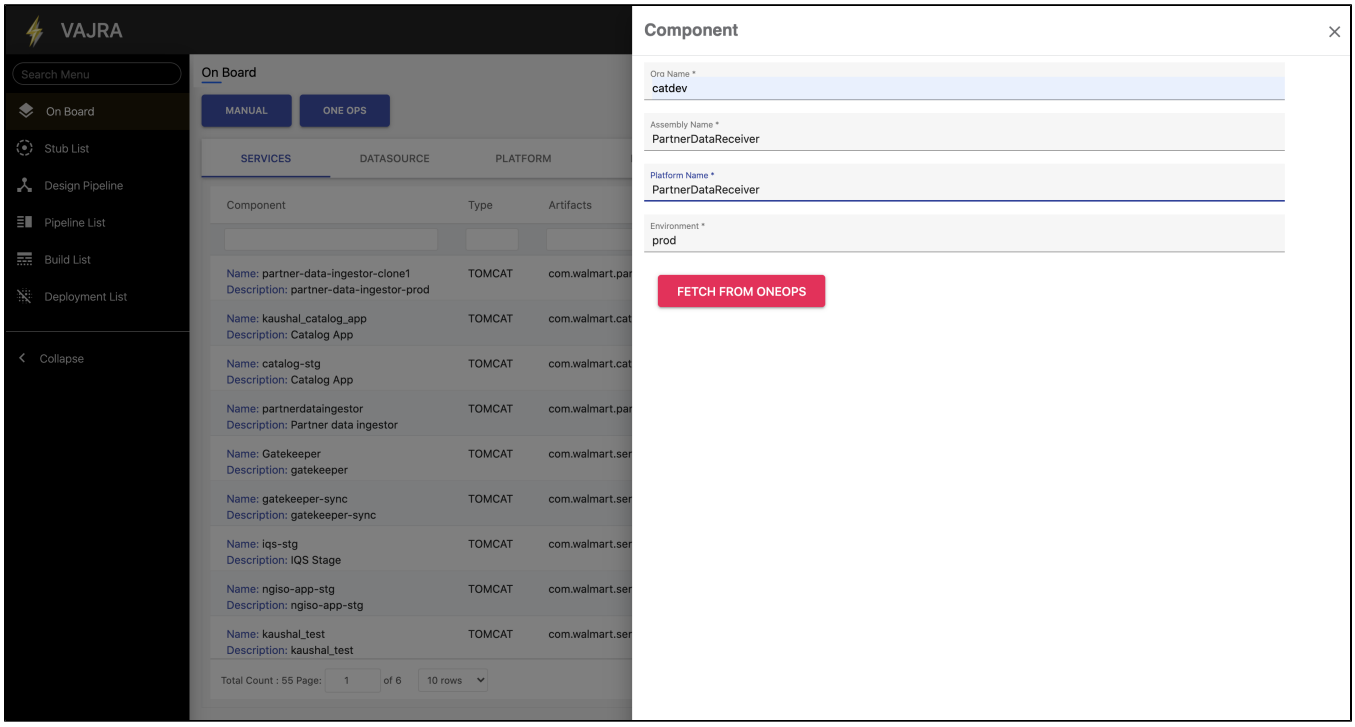
Caution: Please be aware that the HTTPS, HTTP & TCP Whitelist DNS mentioned in the component onboarding will be whitelisted from the sandbox environment of the vajra and components will be able to access it when you run it in Kubernetes environment.

Onboarding through one Ops configuration :-

Applicable for Tomcat & SpringBoot only .

If the user has already one ops assembly then vajra tool can get details from the design and fill in all the information that is needed to dockerize the system. Currently, this will work only for the **Tomcat** & **SPRINGBOOT** design. Please enter the one ops design information like Org Name, Assembly Name, Platform Name, Environment and click fetch "From One Ops button". Vajra tool will get all details from the One Ops and fill in the onboard form for you.

For this importing to work we need access to your org. Please provide access to "svcvajra" user in your org before you import the assembly.

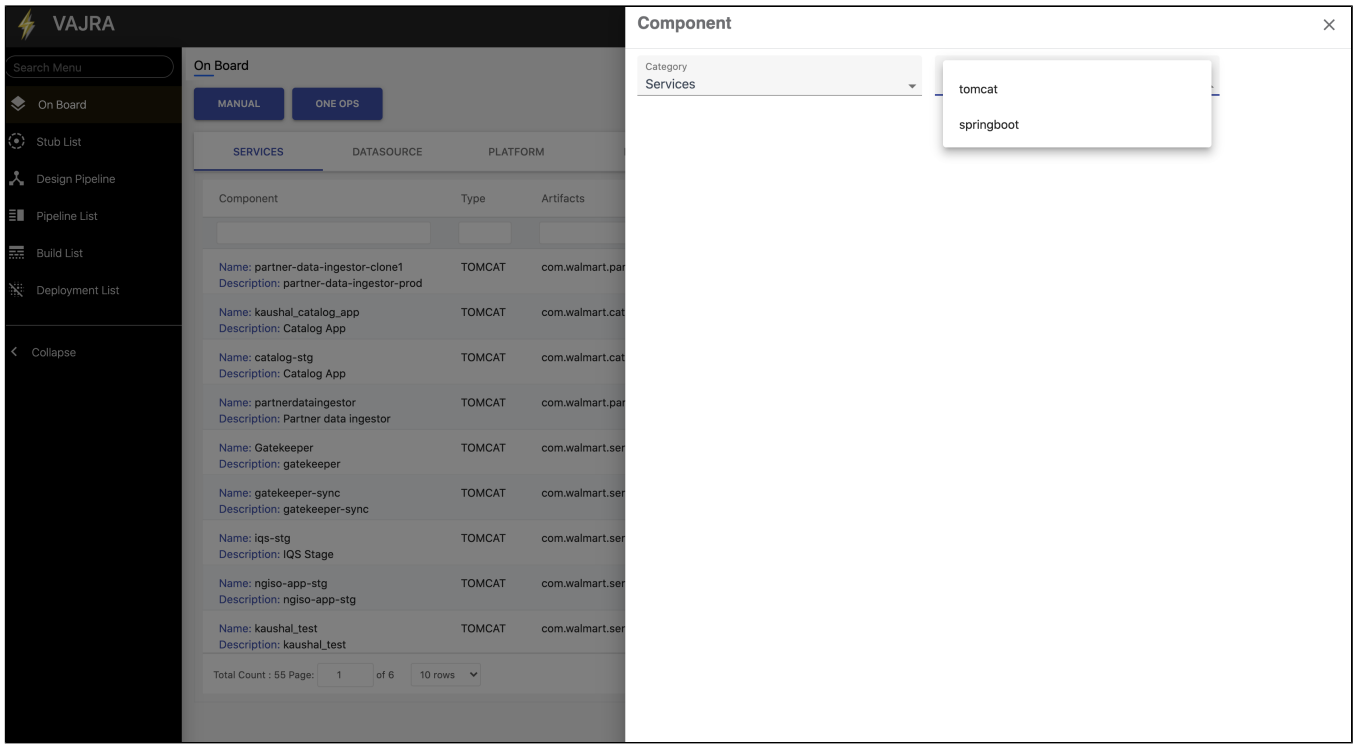


Onboarding through Manual Configuration :-

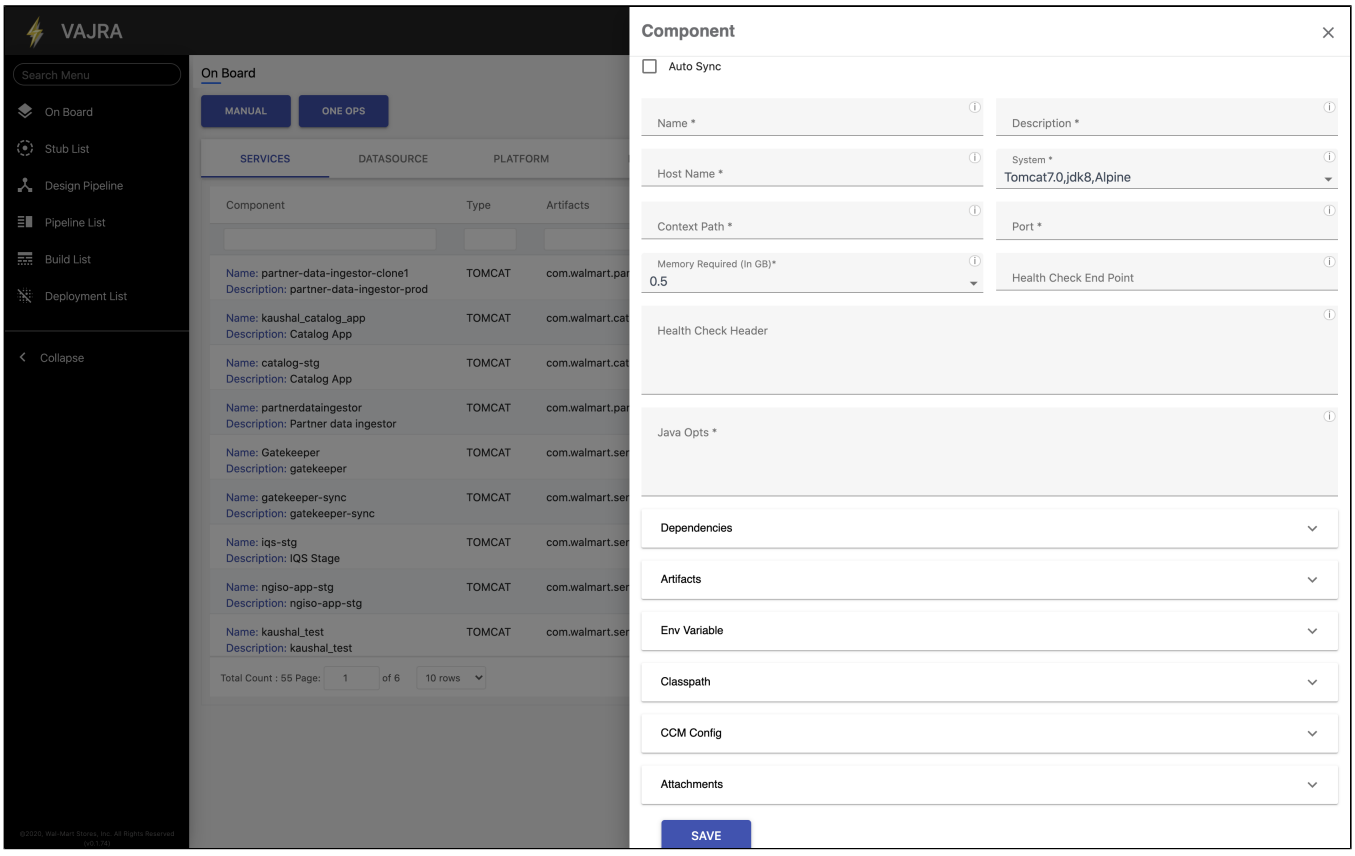
Applicable for all the Components Like Tomcat , SpringBoot , Cassandra , Kafka , Storm & Docker .

1. Tomcat Or SpringBoot Manual Onboarding Steps :-

Click on `On Board` present on left side of screen Click on `SERVICES` tab Click on `MANUAL` button Select `Services` as `Category` & `tomcat /springboot` as `System` in the right panel .

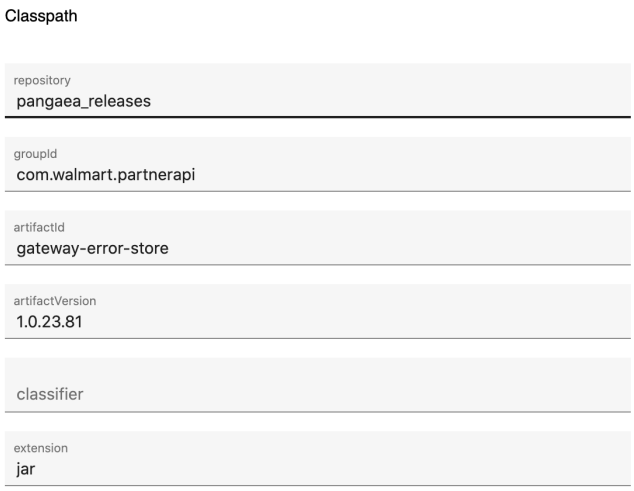
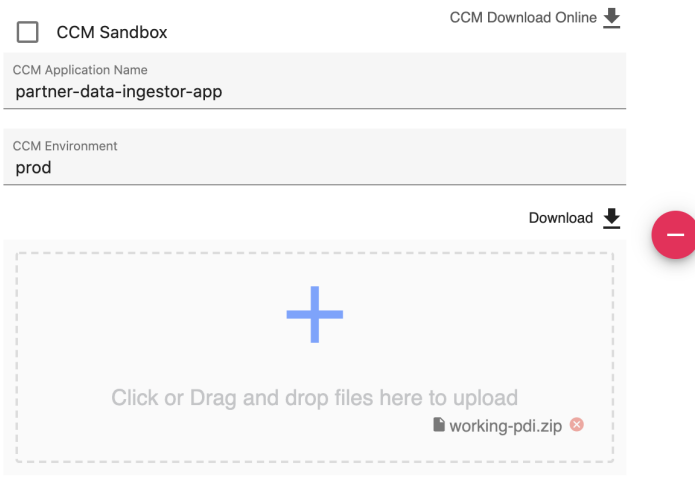


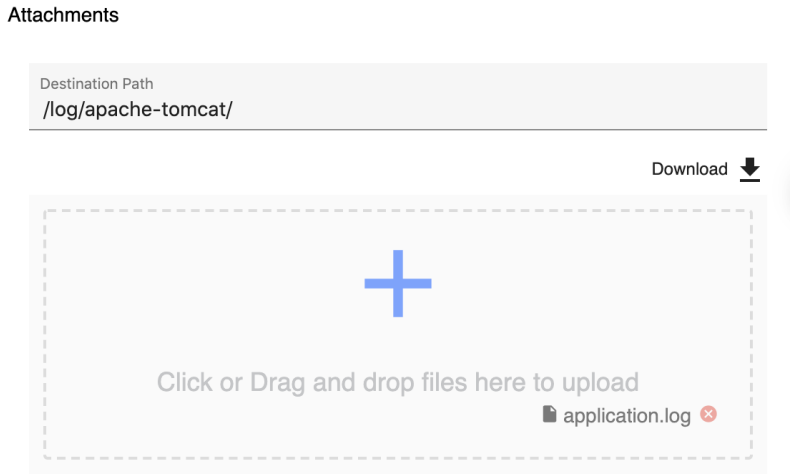
Below is the Tomcat onboarding form. Springboot have similar form to Onboard .



Input Fields required to do Manual Tomcat / Springboot Onboarding :-

Fields	Description
Auto-Sync	Enabling of this check box is required if you want to sync your onboarded component later point of time with nexus repository
Name	Name of your system . Vajra accepts unique name . So better to add your component env suffix after the name, eg- Partner-data-reciever-prod
Description	Description of the component and the system that is represents
Hostname	The Hostname of the system , example- partner.dataingestor.prod.walmart.com
System	It's combination of tomcat, Jdk & Os version, on which your system will run
Context path	It's web app's context path, example- partner-data-ingestor-app
Port	Port on which the tomcat service is exposed .
Memory required	The minimum amount of RAM required for the system to start and take up a few requests
Health Check End Point	It's web app's health check end-point , example - /deepCheck
Health Check Header	Headers required to hit health check end point.
Java Opts	The JAVA_OPTS needed for your application in tomcat
Dependencies	If your system is dependent on another system which is a READ only system then you can mention the hostname of that system in this field.Hostnames that are mentioned here will be whitelisted and will be allowed by the vajra network to contact the actual system in Walmart network. Currently vajra supports HTTPS, HTTP & TCP as Dependencies host . **Note: It is not recommended that you include any production hosts in your dependencies whitelist. Please note that if you add any prod host to the dependencies, it will make calls to your prod services and pollute the prod data.**
Artifacts	It's nexus repository details like Nexus Repository, Group Id, ArtifactId, Extension & Classifier (Optional). Example is shown below . 
Env Variable	Environment variables that need to be set in the system during the deployment

Classpath	If system require any jar to be in classpath, please use it . Example shown below . 
CCM Environment	If you are system is using CCM and if you want to override any of the properties specifically for integration testing please mentioned it in this section. CCM Application Name, CCM Environment and overriden properties file as a ZIP format. Please note all the files has to be selected together and then zip it for upload. CCM Download Online - This utility will help you override and download all properties of the given CCM configuration . There is also an options for CCM Sandbox which will allow you to put all the CCM configuration locally inside the K8 container instead of reading any configurations from CCM server . And you can have more than one CCM uploads as part of your system requirement in vajra . Example shown below .  NOTE :- Zip download options will be available in component edit mode . You can download your uploaded zip file locally for review & editing . Same applicable for attachments also .

Attachments	It allows you to attach any files for writing a logs, reading secret for system etc . Example shown below . 

After the Onboarding process is complete , we can click on the `SERVICES` tab to list down all the onboarded components along with other details like Created_by, Created_On etc . There is **ACTION** drop down list to **view, edit, delete** the onboarded component along with **CLONE & SYNC** features .

CLONE

This feature will allow you to clone a component on just a button click . Example suppose you want to do some changes in the onboarded component but if you do it in existing one it can impact running pipeline. So better first clone it and then do the modification . After that you can add it to the pipeline design for your testing . Currently clone feature is applicable for TOMCAT, SPRINGBOOT, STORM, and DOCKER components.

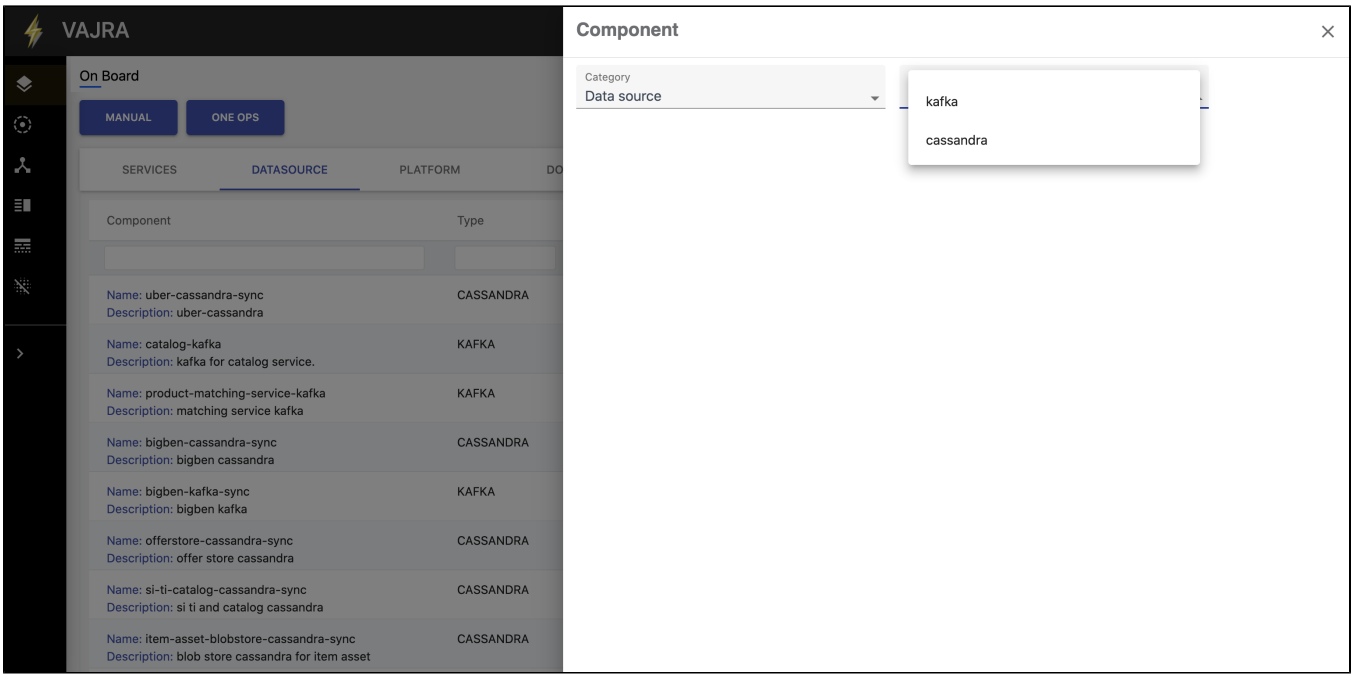
SYNC

This feature will allow you to do sync of your onboarded component with nexus repository . Example suppose some team has pushed new jar/war file to nexus for your onboarded component but your pipeline is still using the old one . So in this case you can sync all the components or the required components involved in the pipeline design and can upgrade the pipeline with all the latest war .

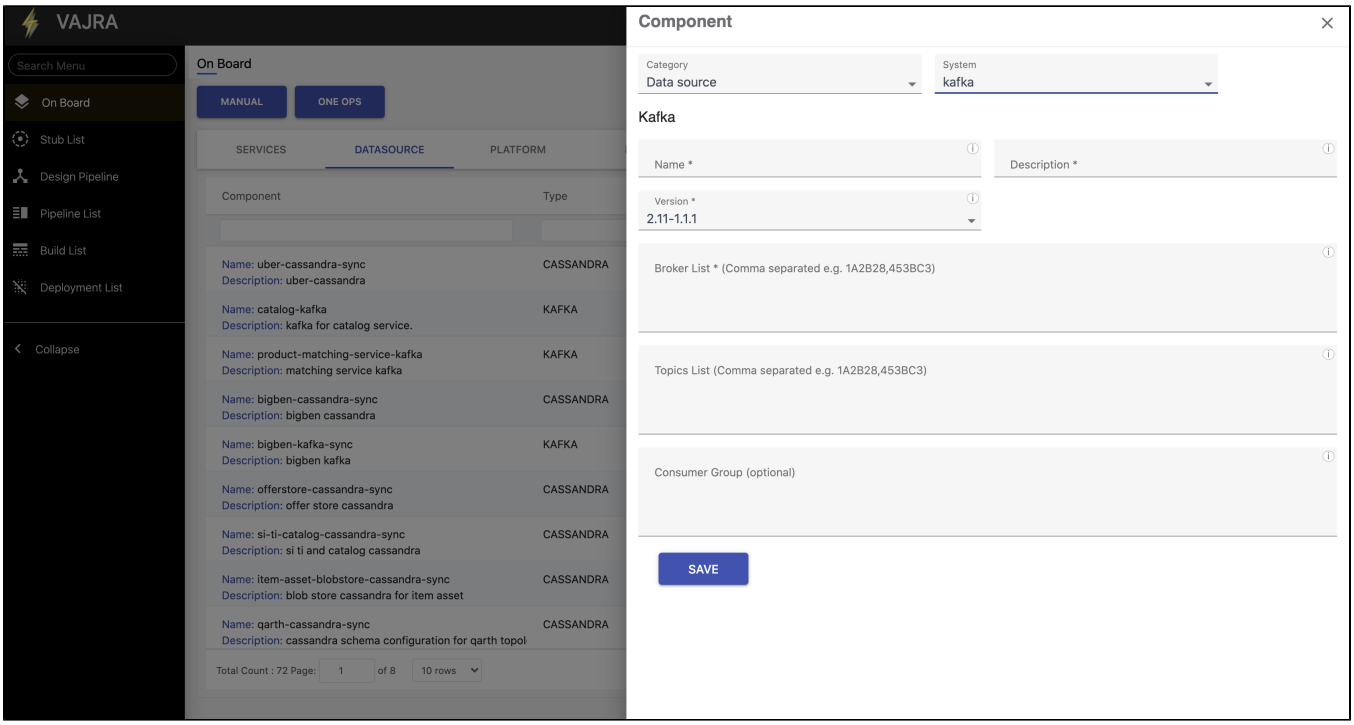
2. Kafka Manual Onboard Steps :-

Click on `On Board` present on left side of screen Click on `DATASOURCE` tab Click on `MANUAL` button Select `Data source` as `Category` & `kafka` as `System` in the right panel .

Note: If you deploy any data sources like Kafka and Cassandra in WCNP cluster for an extended period of time, it may result in problems down the road after a few days in the pipeline because the maximum disc space permitted in containers is 1GB.



#Below is the Kafka onboarding form .



#Input fields required to do Manual kafka onboarding .

Fields	Description
Name	Kafka system name. It is better to attach for which environment you are configuring the component, Eg. feed-gateway-kafka-stage
Description	Description of the component and the system that is represents

Version	Kafka version
Broker List	List of broker hostnames as comma-separated values
Topic List	List of topics to be created. Note: Auto topic creation is enabled by default in vajra
Consumer group	List of consumer groups to be created

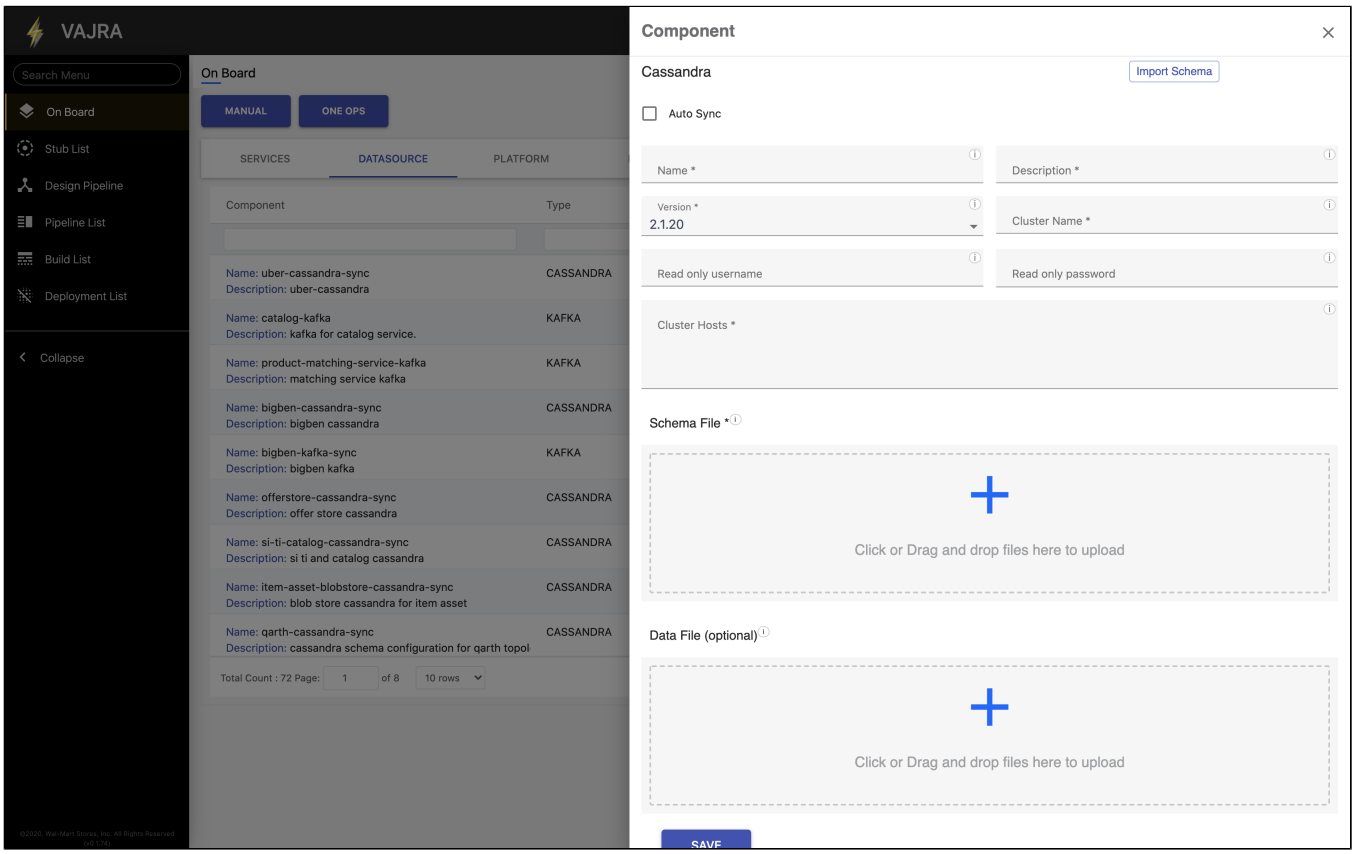
3. Cassandra Manual Onboarding Steps :-

Click on `On Board` present on left side of screen Click on `DATASOURCE` tab Click on `MANUAL` button Select `Data source` as `Category` & `cassandra` as `System` in the right panel .

Please refer step 2 above for screen shot .

Note: If you deploy any data sources like Kafka and Cassandra in WCNP cluster for an extended period of time, it may result in problems down the road after a few days in the pipeline because the maximum disc space permitted in containers is 1GB.

#Below is the Cassandra onboarding form .



#Input fields required to do Manual Cassandra onboarding .

Fields	Description
Auto Sync	Enabling is required for syncing the Cassandra component with latest schema file .
Name	Cassandra system name. It is better to attach for which environment you are configuring the component, Eg. feed-gateway-cassandra-stage
Description	Description of the component and the system that is represents
Version	Cassandra version. Currently supported version is 2.1.20 & 3.11
Cluster name	Cassandra cluster name (e.g. hyperloop)

Read only username	Username & Password of cassandra cluster is required to import the schema or Syncing the schema from cluster automatically .
Read only password	Username & Password of cassandra cluster is required to import the schema or Syncing the schema from cluster automatically .
Cluster hosts	List of cluster hosts as comma-separated values. It can either ip address or domain names.
Schema file	Upload .cql file representing the schema. It is recommended to use the "Import Schema" button shown above in the snapshot to download the schema file. Make sure the script has only one datacenter "cdc" and network policy as "simple strategy"
Data file	If the system needs a master data then CQL file with data can be uploaded here.

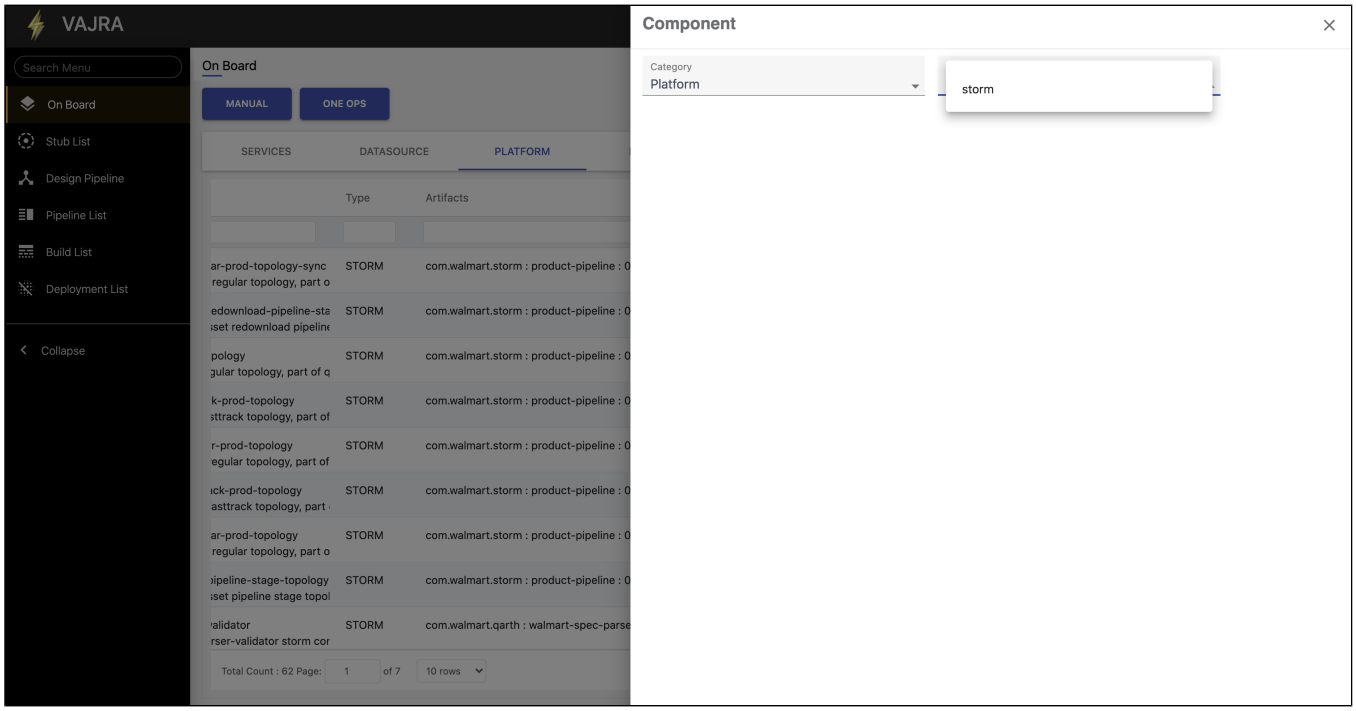
After the datasource (cassandra & kafka) onboarding process complete, We can click on the `DATASOURCE` tab to list down all the onboarded components along with other details like Created_by, Created_On etc . There is also **ACTION** drop down list to **view, edit, delete** the onboarded component along with **SYNC** feature for cassandra .

SYNC

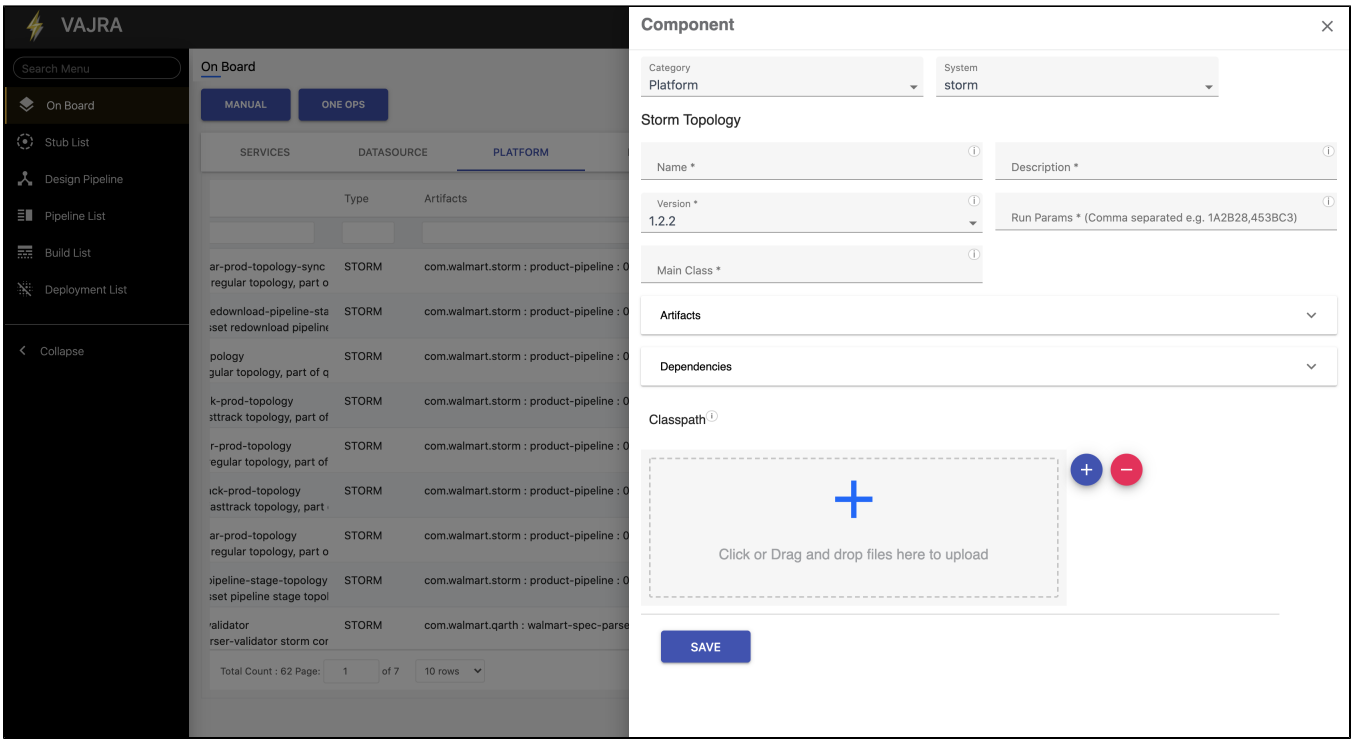
This cassandra SYNC feature will be helpful in upgrading the cassandra schema automatically . But it requires correct username & password to connect cassandra cluster to get latest schema .

4. Storm Manual Onboarding Steps :-

Click on `On Board` present on left side of screen Click on `PLATFORM` tab Click on `MANUAL` button Select `Platform` as `Category` & `storm` as `System` in the right panel .



#Storm onboarding form for Manual onboarding :-

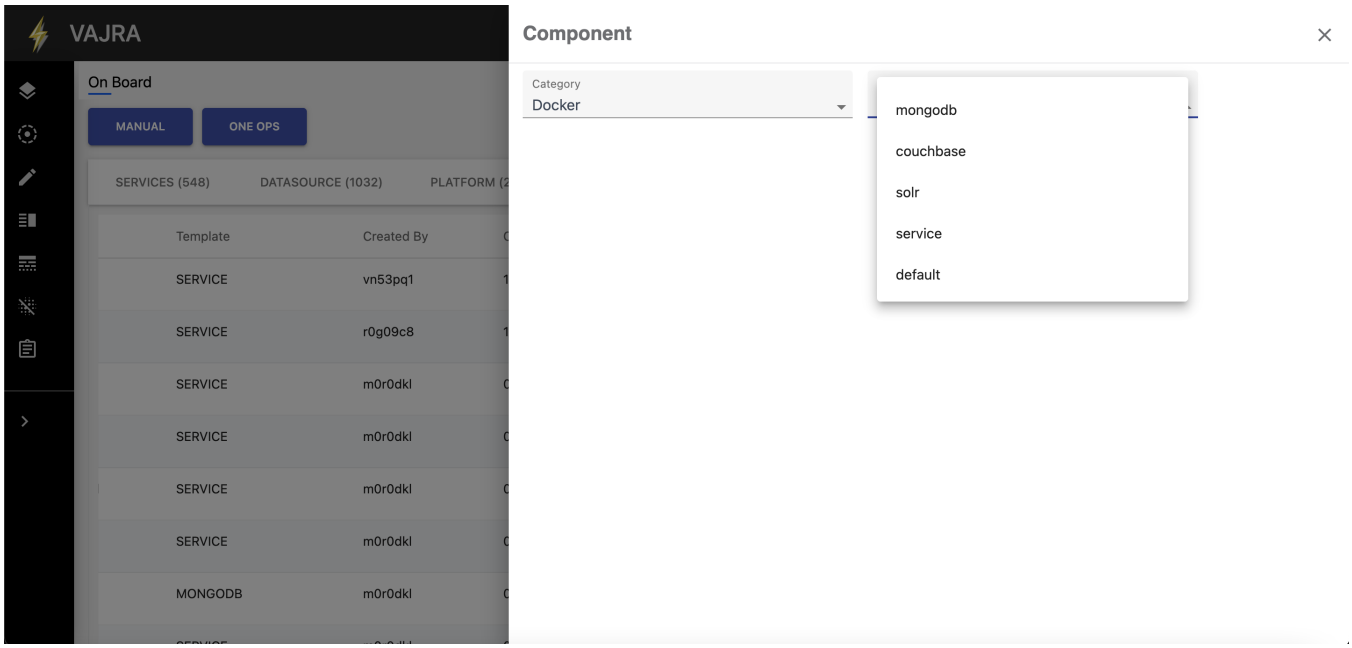


#Input fields required to do Manual Storm onboarding .

Fields	Description
Name	Storm topology name. It is better to attach for which environment you are configuring the component, Eg. spec-parser-validator-stage
Description	Description of the component and the system that is represents
Version	Storm cluster version in which the topology has to be submitted
Run params	The run parameters that need to pass to the submitting topology
Main class	Submitting topology's main class
Artifacts	It's nexus repository details like Nexus Repository, Group Id, ArtifactId, Extension & Classifier (Optional).
Dependencies	If your system is dependent on another system which is a READ only system then you can mention the hostname of that system in this field.Hostnames that are mentioned here will be whitelisted and will be allowed by the vajra network to contact the actual system in Walmart network. Currently vajra supports HTTPS, HTTP & TCP as Dependencies host . NOTE : - Dependencies fields will be in tomcat, springboot, storm & docker . **Note: It is not recommended that you include any production hosts in your dependencies whitelist. Please note that if you add any prod host to the dependencies, it will make calls to your prod services and pollute the prod data.**
Classpath	Add all the files that need to be placed in the classpath of the running topology

5. MongoDB, CouchbaseDB, Solr Manual Onboarding Steps :-

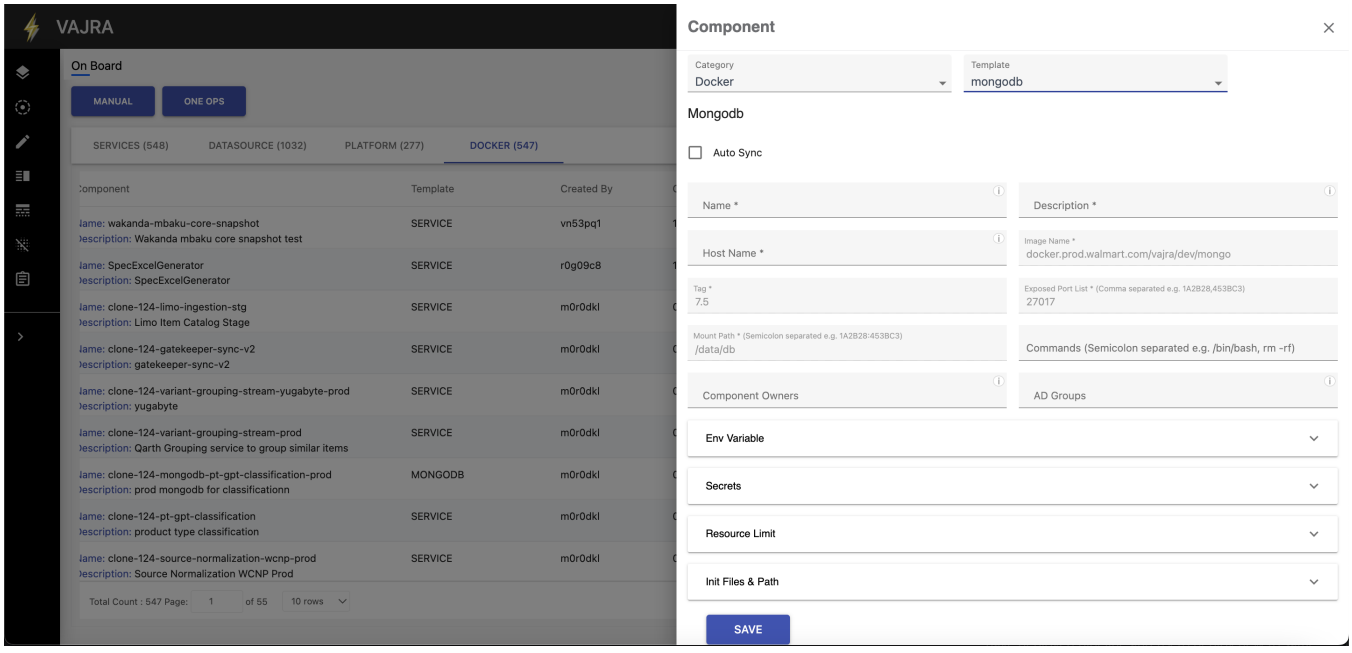
Click on `On Board` present on left side of screen Click on `DOCKER` tab Click on `MANUAL` button Select `Docker` as `Platform` & `mongodb` | `couchbase` | `solr` as `Template` in the right panel .



Please note we have pre-build images for mongo, couchbase & solr Db in vajra . That's why in below snapshot most of the fields values are pre-populated and remaining user has to enter in order to complete the onboarding for those Db.

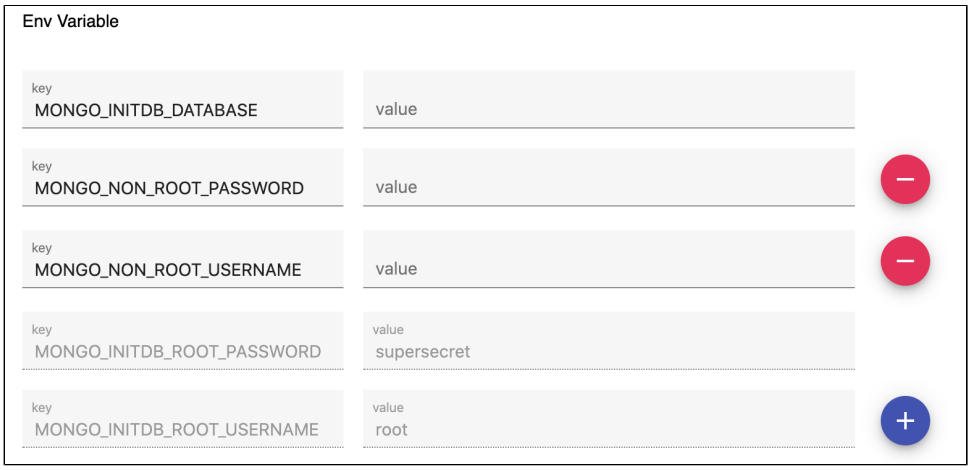
#Supported DB Manual Onboarding form .

NOTE:- Since all have same temple . Hence showing example for one (mongoDb).



#Input fields required to do Manual Db onboarding .

Fields	Description
Name	Name of the System/Db
Description	Description of the system that it represent

Host	Host name / DNS of the Db . Example - mgs-prod.prod.mongodb.cdqarth.prod.walmart.com
Commands	Any specific command needed to run against Db . Like command to load schema for Db
Runtime Params	The run time params if any needed for running the Db
Dependent Hosts	This is currently not used . It has been replaced by `Dependencies` field below
Dependencies	Host name / DNS of any other dependent system . This is optional . **Note: It is not recommended that you include any production hosts in your dependencies whitelist. Please note that if you add any prod host to the dependencies, it will make calls to your prod services and pollute the prod data.**
Env Variable	Environment variables that need to be set in the system for running the Db . 
Akeyless Secrets (If disabling WCNP secrets)	• **srcSecretPath:** The name of the AD-folderName in akeyless Vajra Non-prod folder in which the secrets have been created for by the team. For example, if in akeyless, the secrets have been created in the path '/Non-Prod/vajra/<your-AD-folderName>', then the srcSecretPath must be 'your-AD-folderName' • **destSecretPath:** The path in the pod, where the secrets must be injected. For example, /etc/secrets  To access akeyless and create secrets, please refer here

Akeyless Secrets (If enabling WCNP secrets)	**Note:** Enable WCNP secrets only if this component is used for Header-based routing • **srcSecretPath:** The **full path** in akeyless portal in which the secrets have been created by the team. For example, if in akeyless, the secrets have been created in the path 'Prod/WCNP/homeoffice/test-ad-group/folderName', then the srcSecretPath must also be 'Prod /WCNP/homeoffice/test-ad-group/folderName' • **destSecretPath:** The path in the pod, where the secrets must be injected. For example, /etc/secrets 
Init Files & Path	Placeholder to attach any schema file needed for Db

6. Docker image Manual Onboarding Steps :-

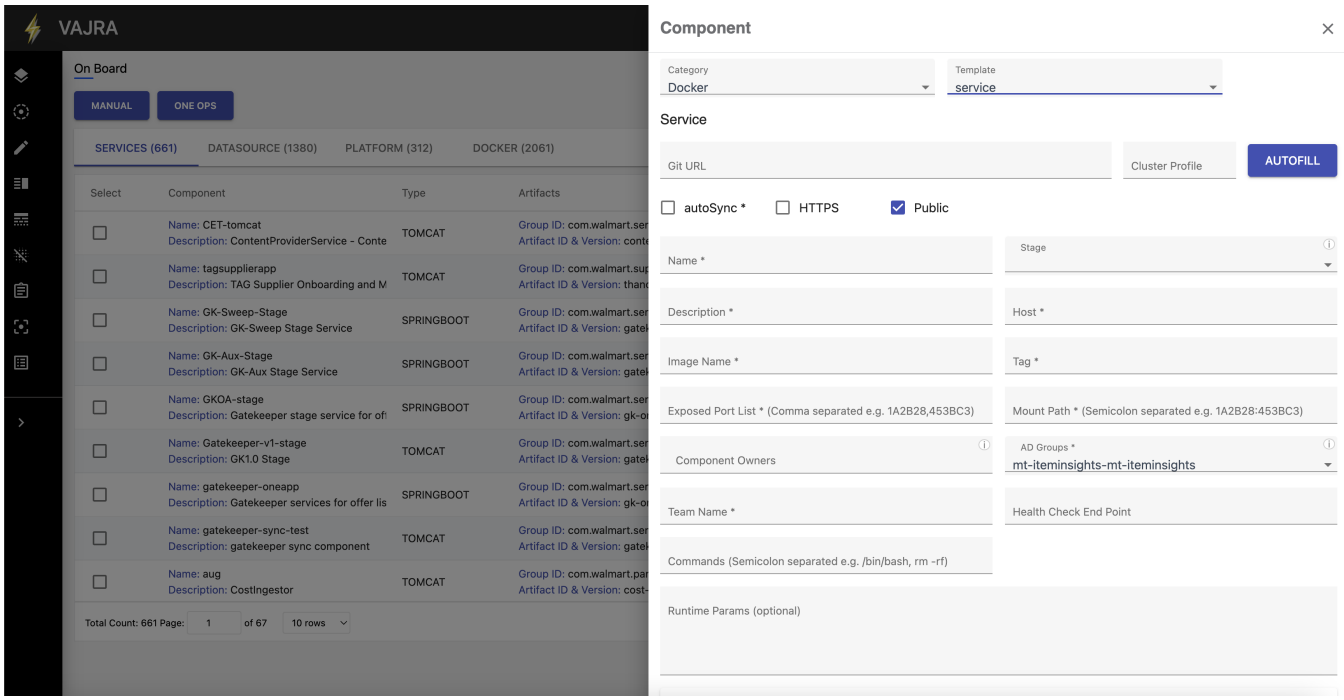
Click on `On Board` present on left side of screen Click on `DOCKER` tab Click on `MANUAL` button Select `Docker` as `Platform` & then select either of the below template

- 'Default' - When you have the docker image and just want to deploy it with the basic functionalities.
- 'Service' - When the docker image you want to onboard is an application/service and want to use additional features like healthCheckEndPoint and CCM config overriding.

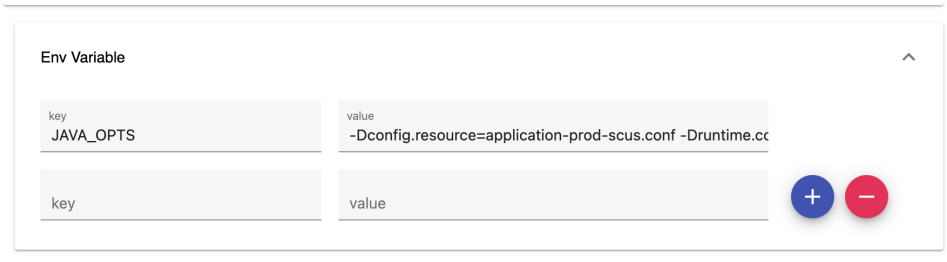
For screenshot, please refer above Docker section.

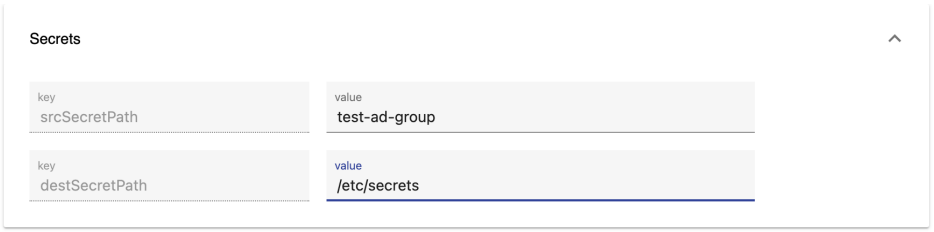
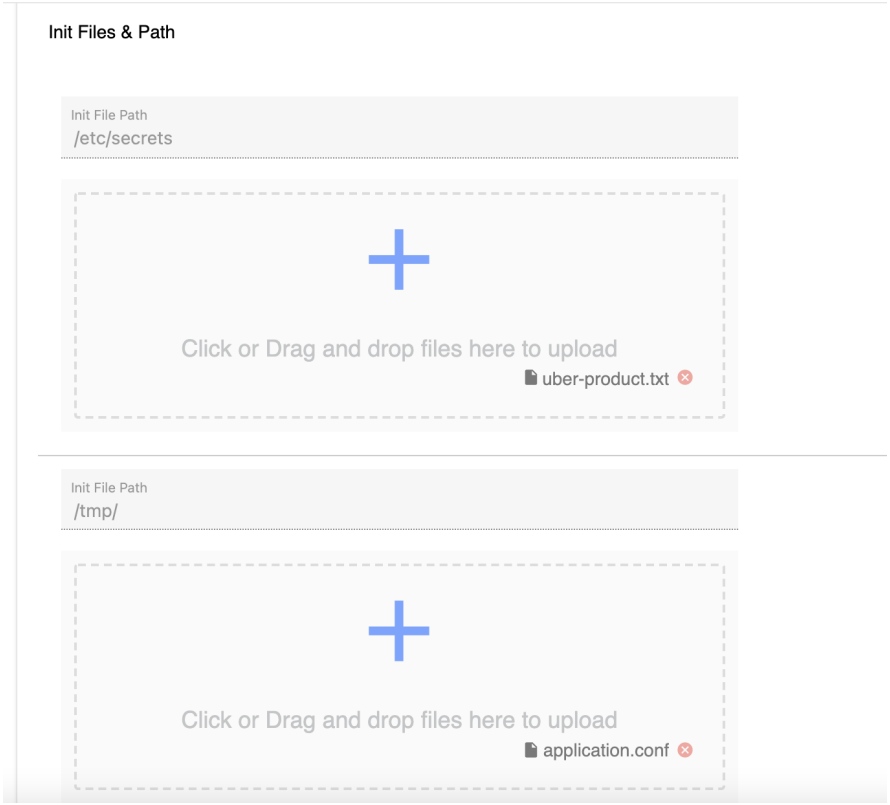
Vajra support onboarding of already build docker image of application . With this we can pass through manual onboarding of system by providing very minimum details in onboarding form .

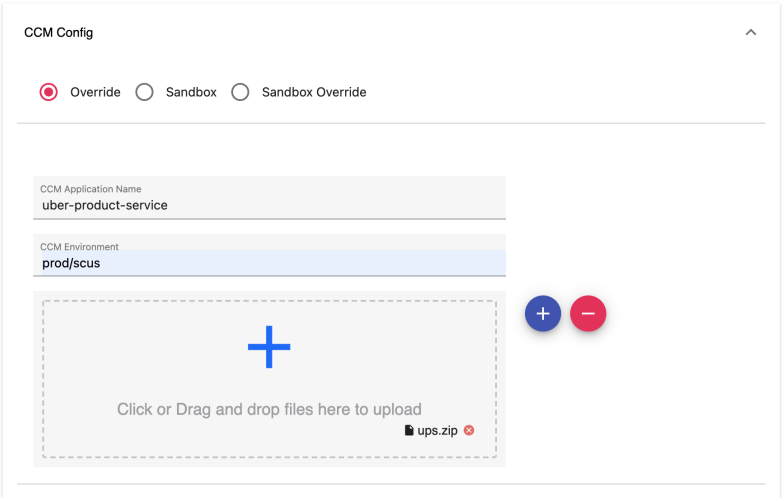
Docker Image manual onboarding form :-

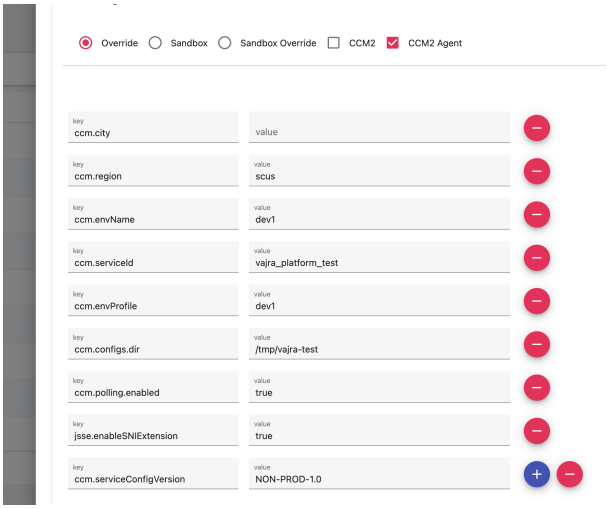


Template is same as Db onboarding shown above except few fields mentioned below . For other fields details please refer above table.

Fields	Description
Name	Name with which you need to onboard the docker component
Image Name	Image path to download from nexus. E.g - docker.prod.walmart.com/catalog-services/uber-slap-akka-stream
Host	The Hostname of the system , example- partner.dataingestor.prod.walmart.com
Tag	Version of the image to be downloaded .
Exposed Port List	Port number exposed by application for accessing it from outside .
Mount Path	Currently this field is not getting used but since made mandatory in UI . Please enter some path like /data/schema. We will remove it .
Health Check End Point	It's web app's health check end-point , example - /deepCheck
Runtime params	The run time params if any needed for running the Db
Dependencies	Host name / DNS of any other dependent system . This is optional . **Note: It is not recommended that you include any production hosts in your dependencies whitelist. Please note that if you add any prod host to the dependencies, it will make calls to your prod services and pollute the prod data.**
Env variable	Environment variables that need to be set in the system for running the Db . 

Secrets	• **srcSecretPath:** The name of the AD-folderName in akeyless in which the secrets have been created by the team. For example, if in akeyless, the secrets have been created in the path '/Non-Prod/vajra/<your-AD-folderName>', then the srcSecretPath must be 'your-AD-folderName' • **destSecretPath:** The path in the pod, where the secrets must be injected. For example, /etc/secrets  To access akeyless and create secrets, please refer here
Init Files & Path	Placeholder to attach any init files needed for running the application. This could include any secrets or tmp files. 

CCM Config	If you are system is using CCM and if you want to override any of the properties specifically for integration testing please mentioned it in this section. CCM Application Name, CCM Environment and overridden properties file as a ZIP format. **Please note all the files has to be selected together and then zip it for upload.** There are three options displayed for configuring CCM properties. ▪ override: if you want to override any of the properties specifically for integration testing. ▪ sandbox: sandbox allow you to put all the CCM configuration locally inside the K8 container instead of reading any configurations from CCM server. ▪ sandbox override: a combination of both override and sandbox. **Please note that for docker onboarding, vajra currently supports only override feature, so please choose override during ccm configuration.** And you can have more than one CCM uploads as part of your system requirement in vajra . Example shown below .  NOTE :- Zip download options will be available in component edit mode . You can download your uploaded zip file locally for review & editing. Same is applicable for Init Files & Path section.

CCM2 Agent Config	If you are using ccm2 agent and you want to download the ccm config from the ccm2 agent, then you have to select the ccm2 agent in Docker onboarding. Please provide the all the parameters in ccm2 agent.  ccm.envName: Add the ccm2 environment for which you want to add the config (you can find this on ccm2 portal https://admin.tunr.walmart.com/) ccm.serviceId: (you can find this on ccm2 portal https://admin.tunr.walmart.com/) ccm.polling.enabled: true jsse.enableSNIExtension: true ccm.configs.dir: Add the config directory ccm.region: scus All the params shown in the screen shot are mandatory. Please don't remove any of them. incase want to override and values wrt ccm2 agent, create a new profile for vajra in ccm2 and then give that profile name here int he configs.

Azure SQL Onboarding

To onboard azure sql instances onto vajra, you can refer to the existing docker component "azure-sql". This component can be cloned and used.
This is azure sql docker image provided by vajra itself. Users are required to provide schema and seed data files alone in the component under Init Files section.

On Board Component

Category: Docker

Template: default

Default

- [ ] autoSync *
- [ ] HTTPS
- [x] Public

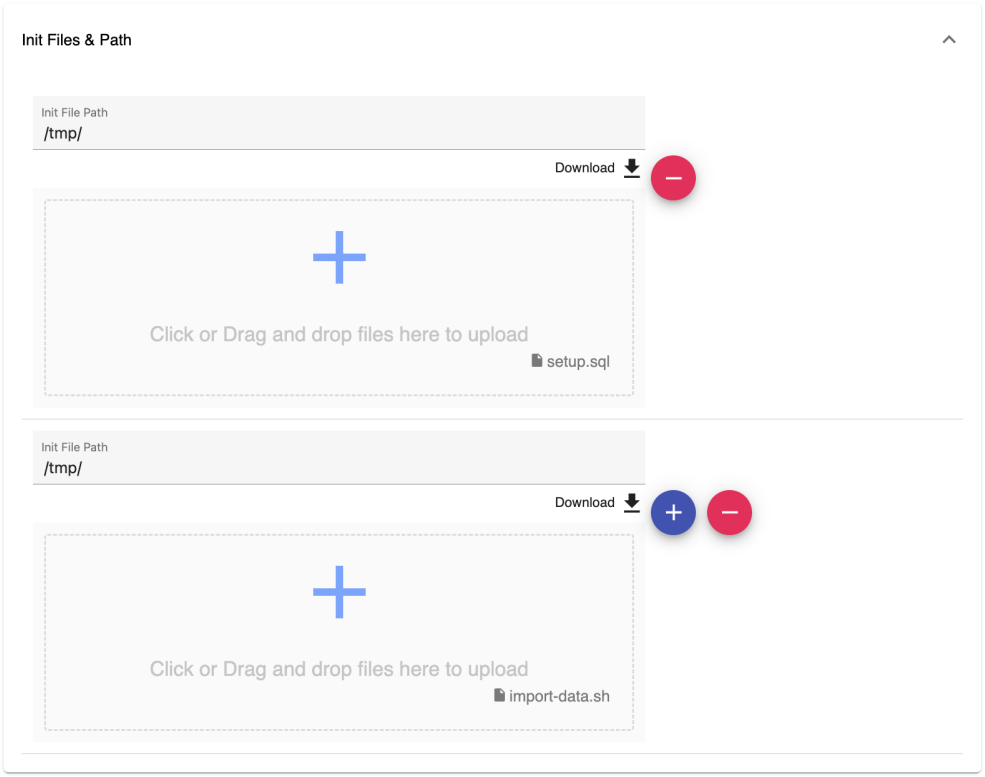
Name *: azure-sql	Description *: testing
Stage: non-prod	Host Name *: azuresql.vajra.com
Image Name *: docker.ci.artifacts.walmart.com/vajra-docker/azure-sql	Tag *: 16.0.4-dev
Exposed Port List * (Comma separated e.g. 1A2B28,453BC3): 1433	Mount Path * (Semicolon separated e.g. 1A2B28:453BC3): /
Commands (Semicolon separated e.g. /bin/bash, rm -rf)	Component Owners: n0k00we,b0u002n,b0b03iu,m0k08yh
AD Groups *: vajra	Team Name *: vajra

Env Variable

Akeyless Secrets

Resource Limit

Init Files & Path



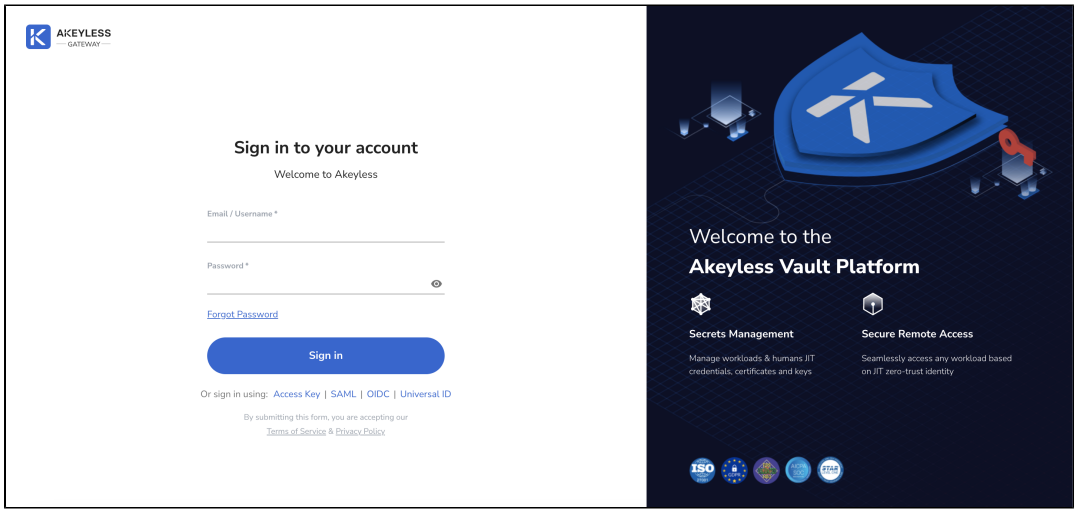
Akeyless and secrets Injection:

Several applications have secrets and would want those secrets to be injected into the pod at a particular path. Hence, Vajra leverages a platform called akeyless to store secrets inject them into vajra pods during deployment. **Users can reach out to Vajra team for AD access to Akeyless**. Once they do, they must be able to create their secrets inside the root directory **'Non-Prod/vajra/<your-AD-folderName>/'**.

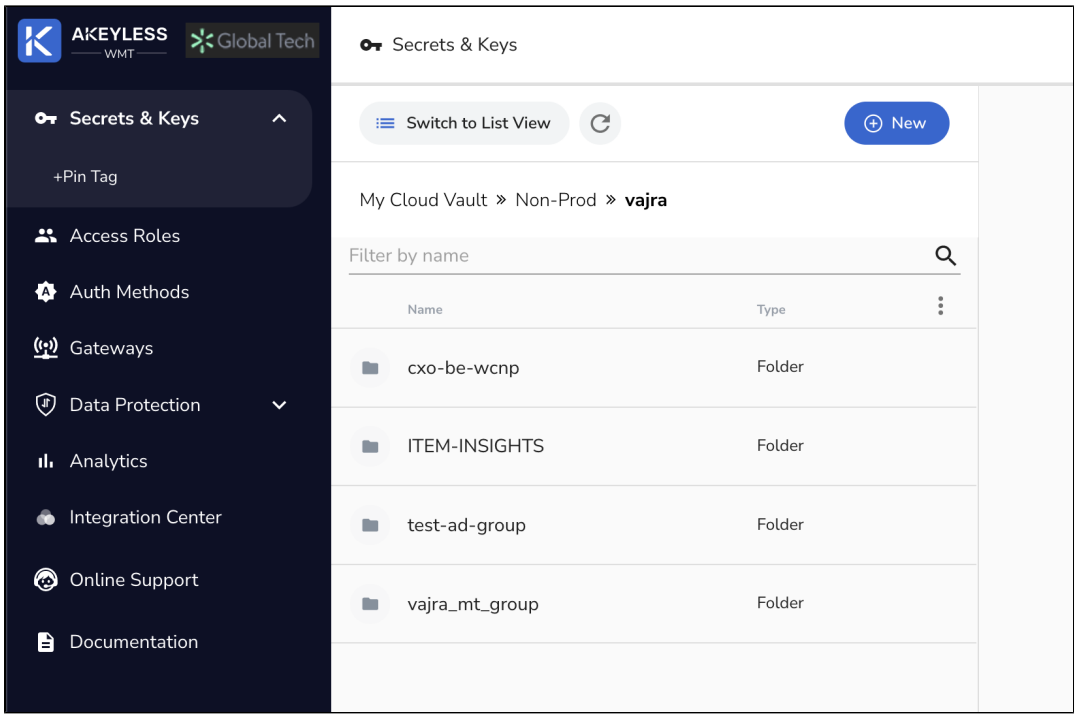
Please follow the detail steps on how to access akeyless and creating secrets given below.

- Please raise a JIRA request to get access to Vajra akeyless. Users can raise access request here

- Login to akeyless portal using your LDAP credentials.

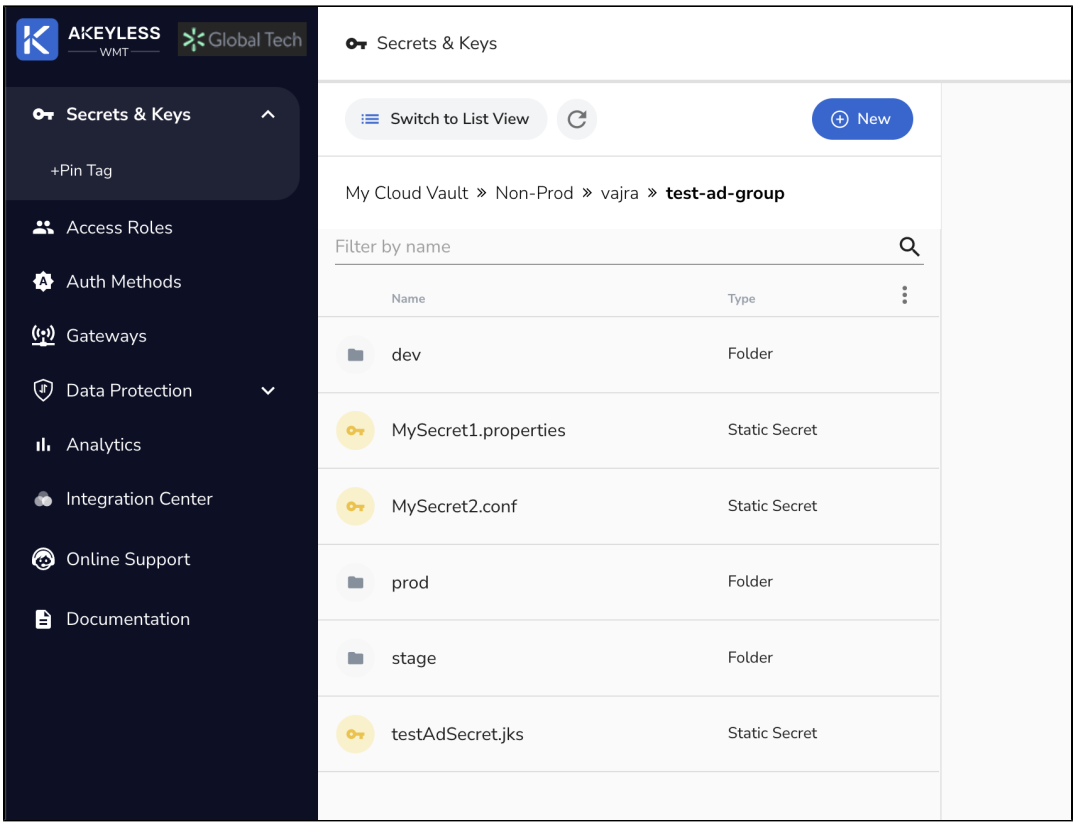


- User will now be able to see his AD-folderName created for the team, once they log in.



- The team can then create their secrets inside that particular folder. Only users who are part of that AD group will be able to view/edit/delete secrets inside that folder.

NOTE: *During deployment we inject those secrets as a file inside the pod. To ease the load on users to provide file extensions for each secrets while onboarding on Vajra, it is expected from users that* ***when you create secrets in akeyless, please provide the file extension as well in the secrets name****. We will be using the secret name as the fileName while injecting secrets into the pods. For eg, if the secret name is 'testSecret' and in path they want to save it as 'testSecret.properties', please save the secret as 'testSecret.properties' in akeyless. Please refer the below example for the same.*



Onboarding Secure Kafka To Vajra

Some applications might be using secure kafka(port-9093) in their applications. Kafka instance provided by Vajra is 9092. Follow these simple steps while on boarding your kafka component in vajra.

1. change port 9093 to 9092. (Override this port in ccm or wherever its configured for your application to connect.)
2. security.protocol to PLAINTEXT from SASL_SSL (Override this config as well wherever its configured)
3. disable ssl from the configs (Override this config as well wherever its configured)

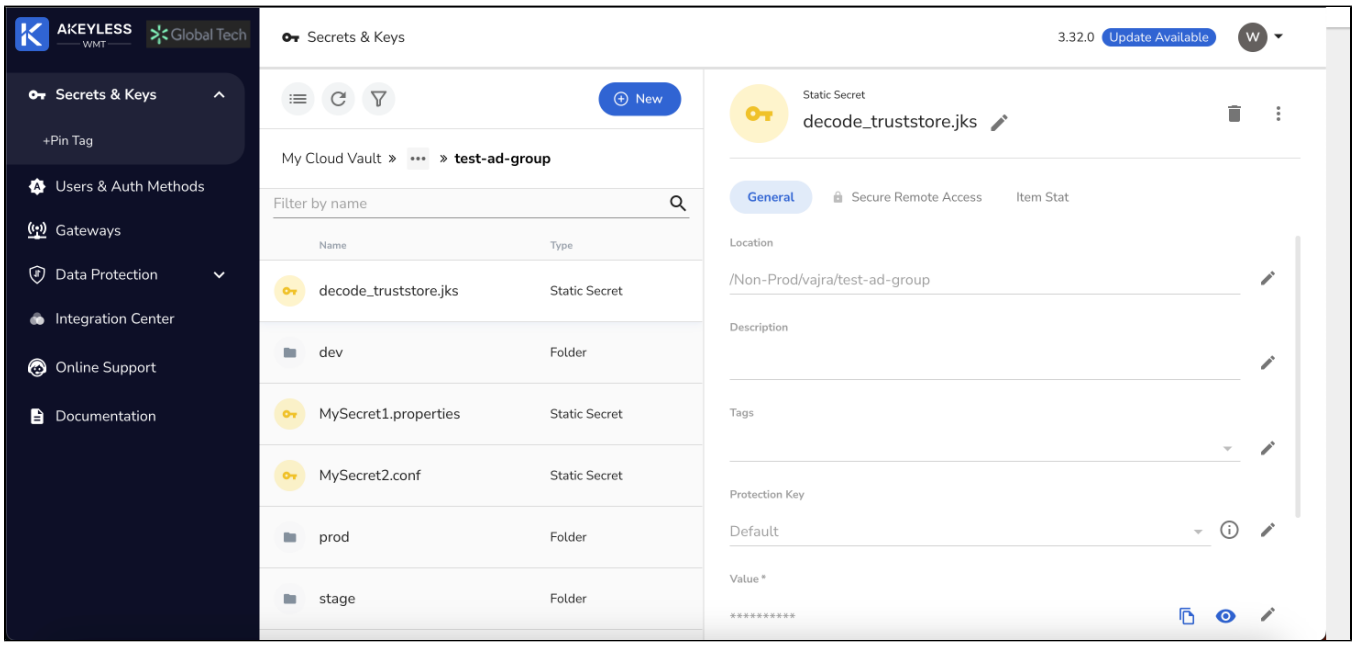
UPLOADING JKS FILES IN AKEYLESS:

Currently, Vajra only supports static secrets to be injected during pod deployments. Hence, it is not possible to directly upload the jks files. Even copying the content of jks file as a secret value would corrupt it. To upload a jks file as a secret, please follow the following steps.

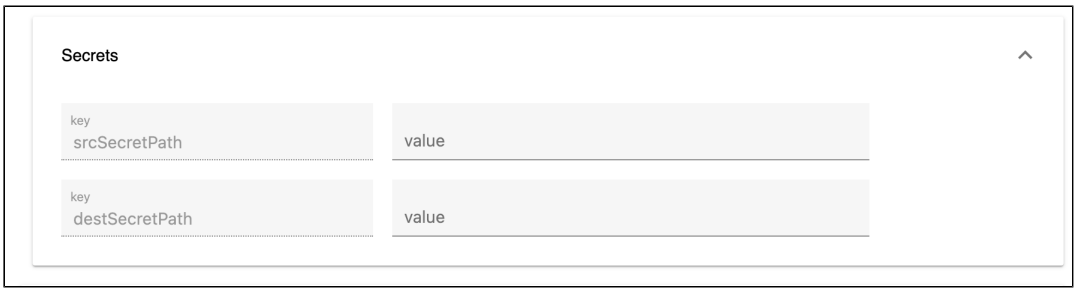
- Encode the jks file in your system using Base64. For example, if the fileName is 'truststore.jks', then run the following command (macOS) in your terminal to encode it. The below command will encode the jks file and save it into a new text file

Command
`base64 -i truststore.jks -o truststore.txt`

- Once the file is encoded, copy the contents of the text file
- Create a static secret in Akeyless with the name '<yourFileName>'. For example, if the fileName is 'truststore.jks', then create a secret where -
  secretName - truststore.jks
  secretValue - copied contents of the encoded text file
- When Vajra injects the secrets into the path, it will automatically decode the secret and have it in place. Please refer the below for an example with a jks file as a secret.



- Once secrets have been successfully created in akeyless, users must provide the secret path while onboarding the component.

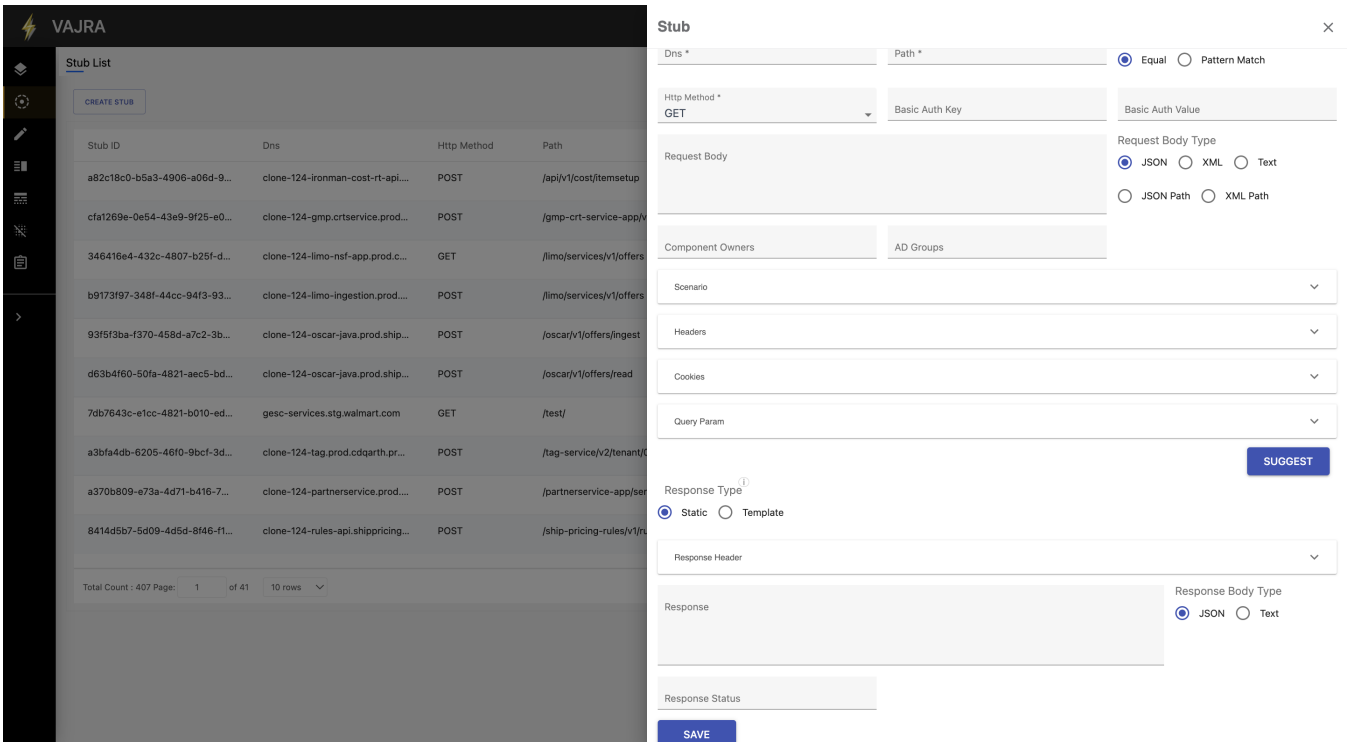


- **srcSecretPath:** The name of the AD-folderName in which the secrets have been created by the team. For example, if in akeyless, the secrets have been created in the path '/Non-Prod/vajra/<your-AD-folderName>', then the srcSecretPath must be 'your-AD-folderName'. Add 'decode_<yourFileName>' in the path to decode base64 encoding.
- **destSecretPath:** The path in the pod, where the secrets must be injected. For example, /etc/secrets

Mocking Or Stubbing the Request

If you don't want to onboard the system or white list it then we have the option of stubbing the component. Please note currently we support STATIC & TEMPLATE as response Type only. Internally we use Wiremock server APIs to mock the req/res , So please refer wiremock docs - http://wiremock.org /docs/request-matching/ as reference guide for onboarding the stub in vajra platform .

Click on `Stub List` present on left side of screen Click on `CREATE STUB` tab Will open Stub onboarding form - shown below for reference

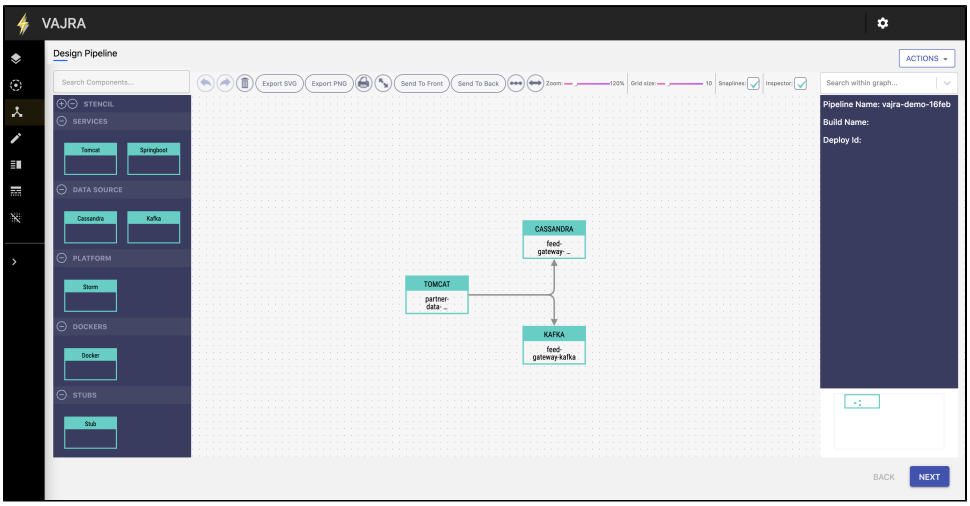


Designing a Pipeline

After all the onboarding are completed we can start pipeline design phase where user will be designing the whole system architecture by connecting the onboarded components .

OPTION_1

1. Click on the "System Design" screen from the left menu.
2. Left panel section (called STENCIL) has list of the components on boarded and stubs
3. In the STENCIL section we have facility to search for components across all section like services, datasources, platform, Docker, stubs .
4. Drag and Drop the component in to the canvas in the middle section . User can also use/drag only the respective template and then search it on the right side panel for the component.
5. Connect the components based on the start up dependency .
6. Before saving the pipeline user can search for any components added through right corner search option during review / validation of designed pipeline .
7. Then go to NEXT and Save the pipeline design after reviewing all the pieces related to pipeline like WHITELIST, SERVICES, DATASOURCE, PLATFORM, STUBS in the SAVE page .
8. After saving the pipeline user have options to BUILD & DEPLOY from that page itself or they can do it from Pipeline List / Build List Page .



OPTION_2 - RECOMMEND TO USE ABOVE OPTION ONLY AS IT IS ONLY FOR UNDERSTANDING PURPOSE .

Click on `Design Pipeline` present on left side of screen It will open design panel along with all the onboarded components and stubs in vajra Drag and Drop the component in to the canvas in the middle section Connect the components based on the start up dependency CLICK on Next .

#Sample screenshot shown below is for designing a pipeline . In the example panel shown below we have following features like :-

a. User can do filtering of their components from the list using the **FILTER** option .

b. User can do **Zoom-in/out** as well while designing a pipeline to get a better view of it in case of bigger pipeline .

c. User can select components and drag it to the right panel for designing .

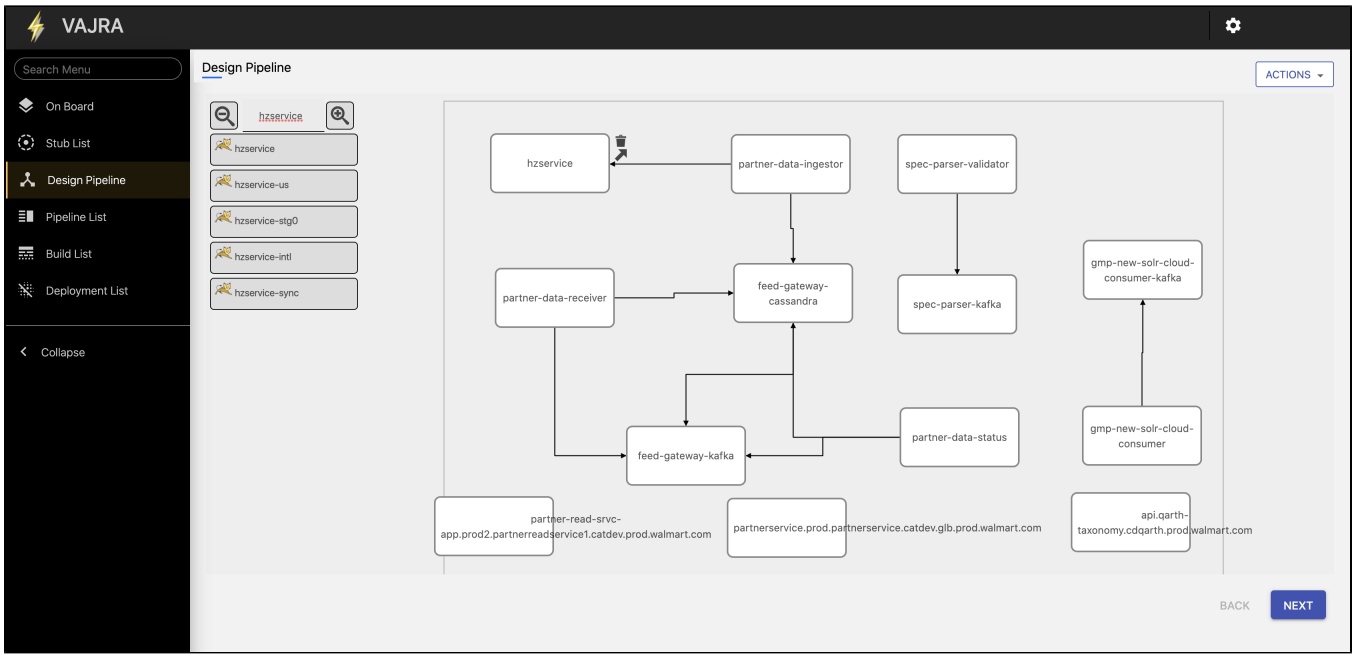
d. On dragged component user have two options - **delete symbol** for deleting the component & **arrow symbol** for connecting to the target component .

#Please note below points while designing :-

a. During design phase right side **ACTION** drop down list will be disabled .

b. Connecting between DATASOURCE to DATASOURCE, DATASOURCE to SERVICES, DATSOURCE to PLATFORM is invalid .

c. At the bottom there are three isolated components which are not connected to anyone represent **STUBS** involved as part of pipeline .

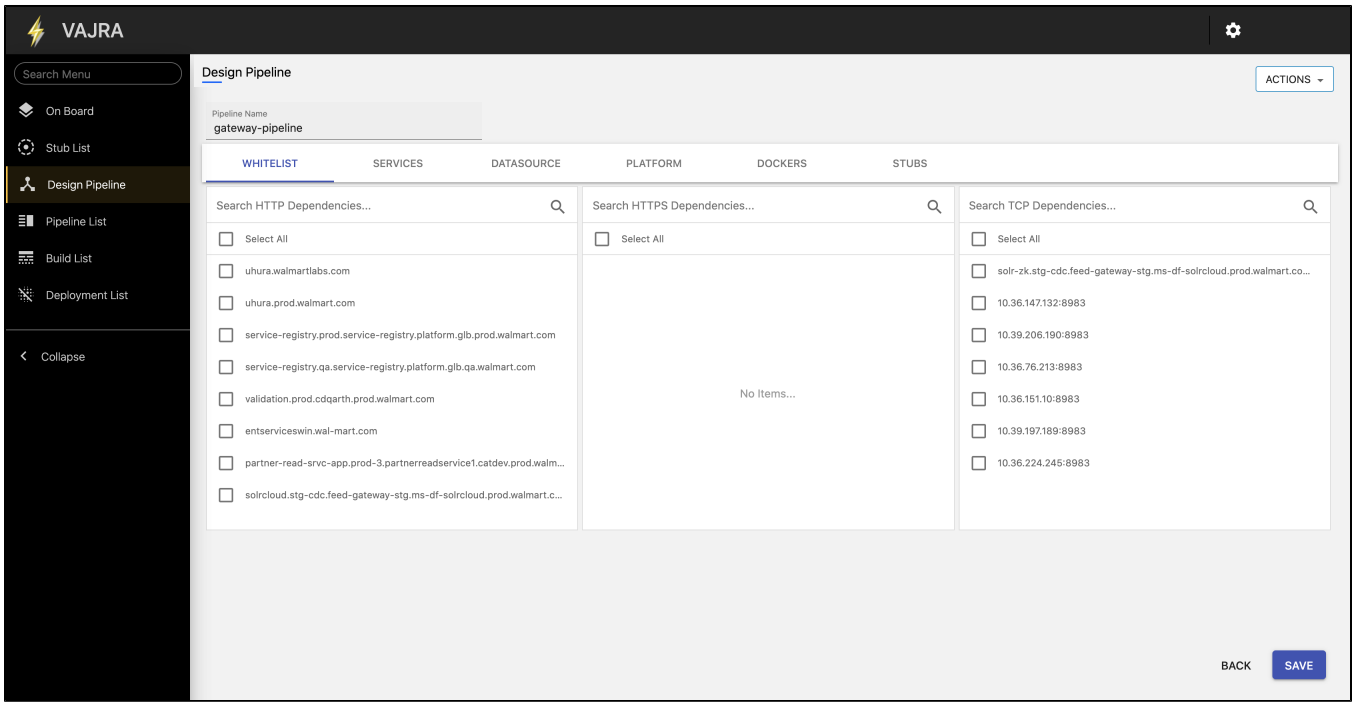


After designing of pipeline is completed please click on NEXT button which will land you to pipeline SAVE page shown below Please enter the name of the pipeline before saving .

On the same page user have two options also as discussed below :-

a. User can click on **WHITELIST** tab to **Select/Unselect** the dependencies (HTTPS, HTTP, TCP) which has been entered as part of component onboarding. So here it will list all the dependencies but as per pipeline requirement user can select ALL or unselect any.

b. Similarly there are TABS to list down the components involved in the pipeline like SERVICES, DATASOURCE, PLATFORM etc .



Building a Pipeline

After the designed pipeline is saved you can find all the saved pipeline in Pipeline List screen as shown below . There are **Actions** & **Build/Deploy Actions** drop down list, discussed in details below .

Actions support below operations on designed/saved pipeline :-

a. **View** - User can view their saved pipeline.

b. **Edit** - User can edit their saved pipeline .

c. **Clone** - User can clone the existing pipeline and save it as new pipeline. You can modify this pipeline as needed.

d. **Delete** - User can delete the pipeline .

Build/Deploy Actions support below operations on designed / saved pipeline :-

a. **Build** - User can trigger a build for the saved pipeline.

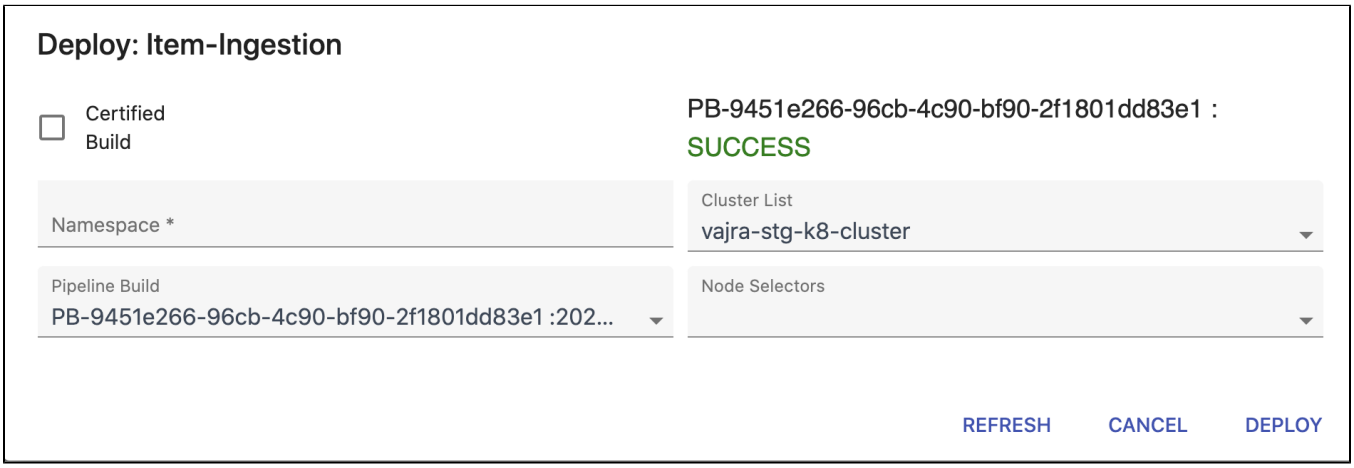
b. **Force Build** - User have option to trigger a force build as well .

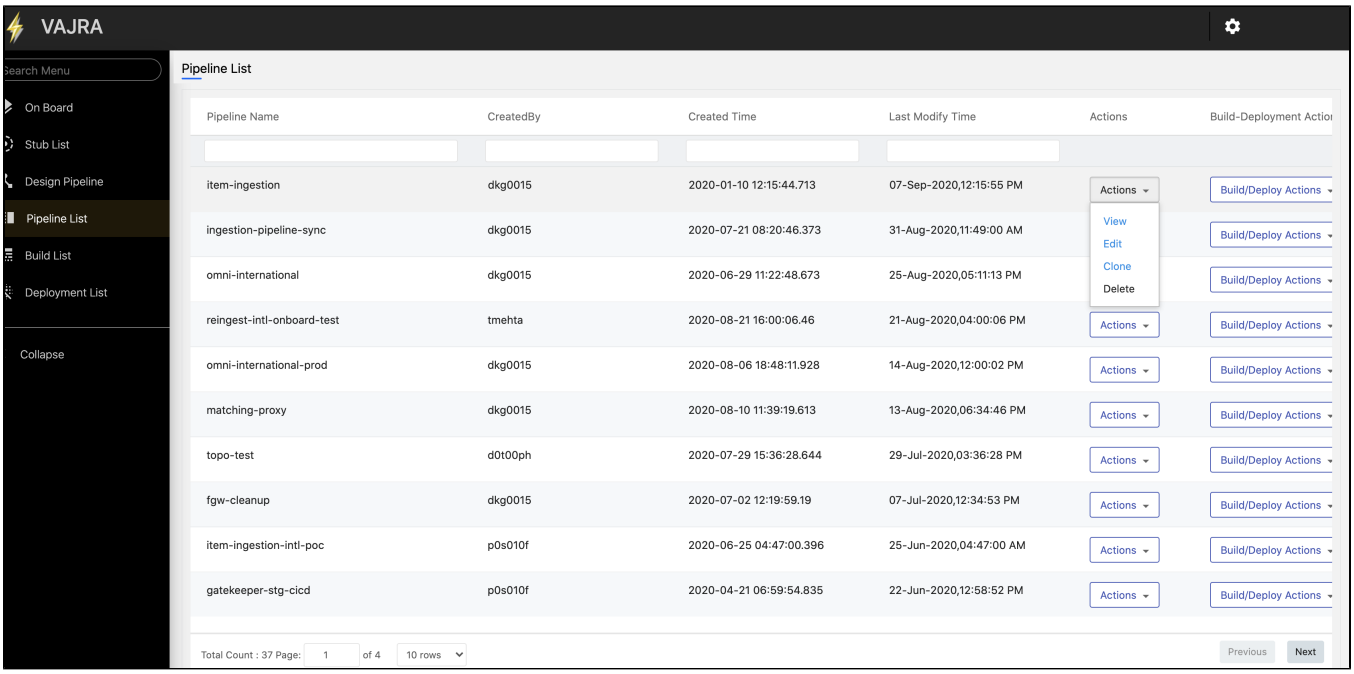
c. **Deploy** - Once build is ready, you can deploy it to kubernetes from here. Enter namespace & select Pipeline build, Cluster, Node Selectors from dropdown and then click on **DEPLOY** button. You can also check the Certified Build options to certify the build . Once the deployment is triggered - vajra will give you concord log url to check the deployment related logs also .

d. **Latest Build Status** - It will show the build status of recent/latest build triggered .

e. **Sync** - If required user can sync their TOMCAT & CASSANDRA components with this options .

f. **Active Deployment List** - It will show the active deployment of all the builds for the pipeline

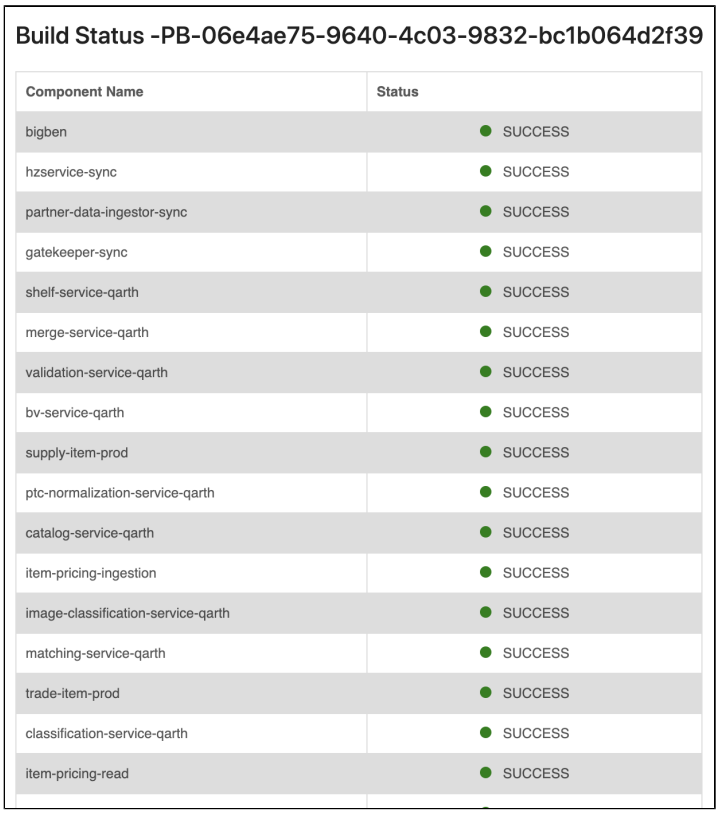




After the build is triggered for a pipeline , We can get list of all the pipeline's build form **Build List .** On the Build List page there is Actions dropdown list provided for some useful features discussed below .

Deploy - User can deploy the pipeline build from here as well . Inputs required to deploy already discussed above .

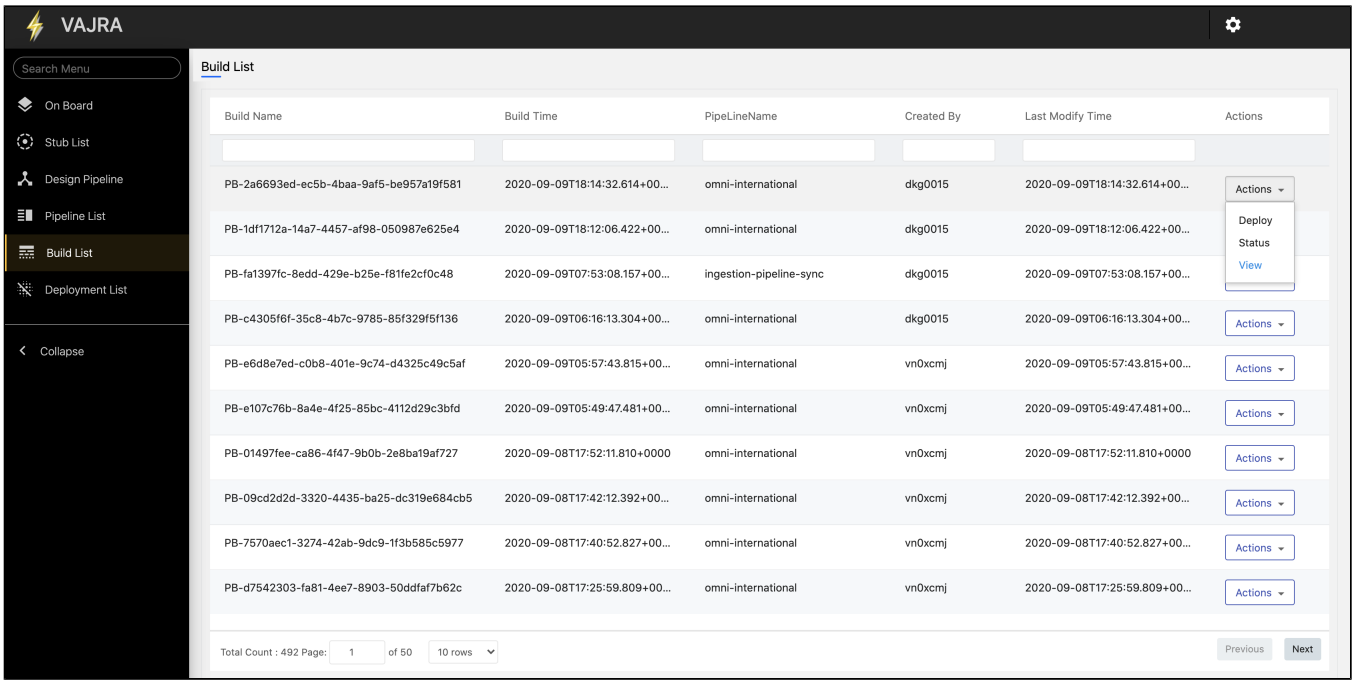
Status - User can view the Pipeline Build status through this option . It's actually the health status of each components present in pipeline .



Build Status -PB-06e4ae75-9640-4c03-9832-bc1b064d2f39

Component Name	Status
bigben	SUCCESS
hzservice-sync	SUCCESS
partner-data-ingestor-sync	SUCCESS
gatekeeper-sync	SUCCESS
shelf-service-qarth	SUCCESS
merge-service-qarth	SUCCESS
validation-service-qarth	SUCCESS
bv-service-qarth	SUCCESS
supply-item-prod	SUCCESS
ptc-normalization-service-qarth	SUCCESS
catalog-service-qarth	SUCCESS
item-pricing-ingestion	SUCCESS
image-classification-service-qarth	SUCCESS
matching-service-qarth	SUCCESS
trade-item-prod	SUCCESS
classification-service-qarth	SUCCESS
item-pricing-read	SUCCESS

View - User can view the graphical representation of their pipeline build along with components build status in the graph .



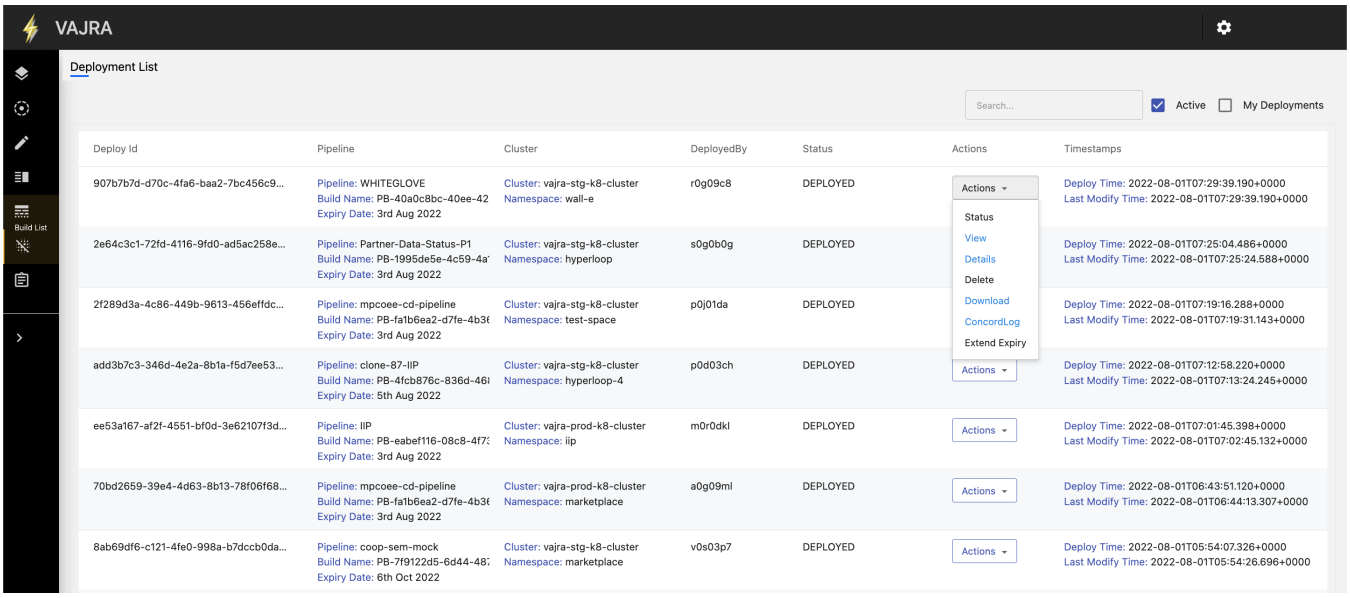
Deploying a Pipeline in K8

Now we are in last stage of Pipeline lifecycle i.e deployment of build pipeline . As we already discussed above we can do deployment from either **DEPLOY** options present in **Actions** dropdown list of `Build IIst` Or **Build/Deploy Actions** dropdown list of `Pipeline List` .

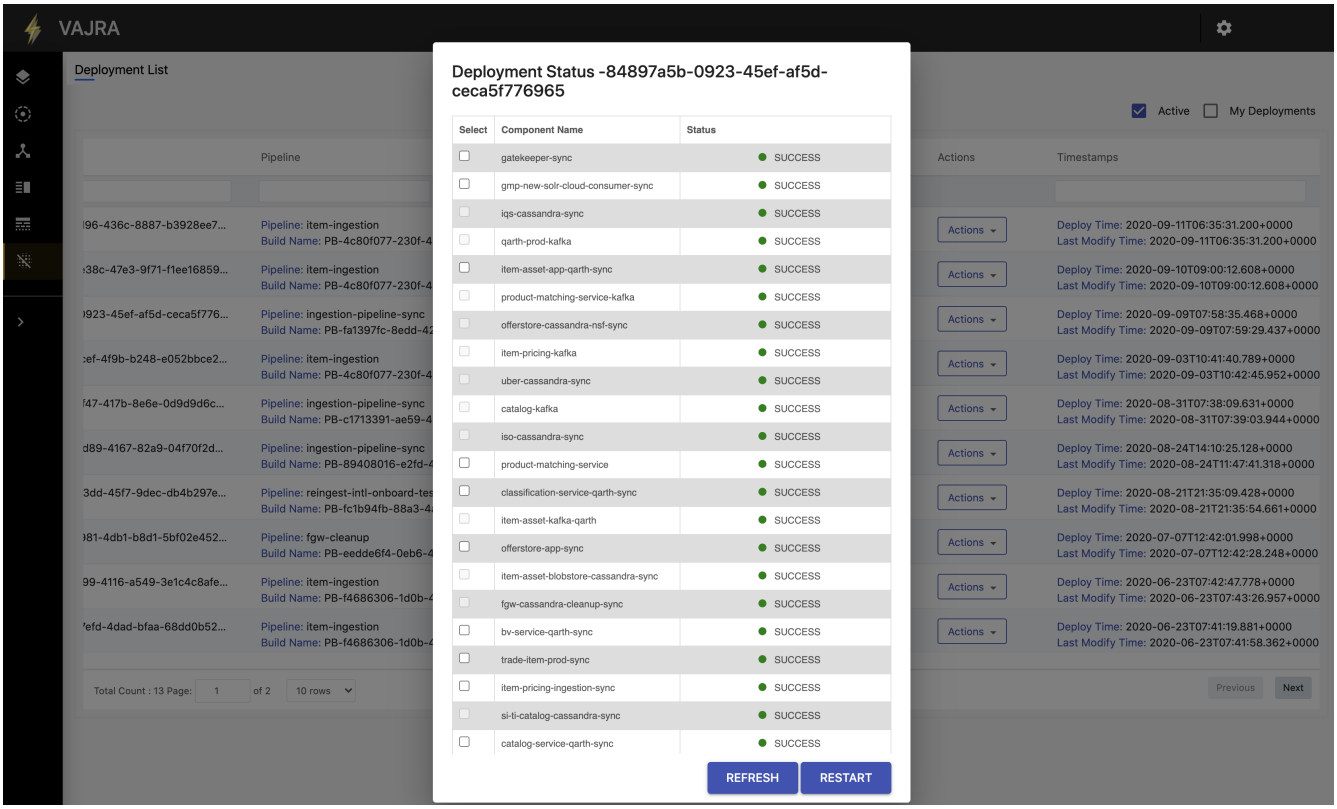
NOTE: The default expiry of deployments done through Vajra is **two days**. After two days, the deployment is deleted to clear up the resources. Although, users can extend the expiry of the deployment if they want to keep the pipeline running for certain period of time. Please refer below the steps to extend the expiry.

Sample screenshot shown below . Which has also Actions dropdown list with some important features discussed below :-

Extend Expiry - As discussed, the default expiry of deployments done through Vajra is **two days**. To extend expiry, please navigate to Deployment List Actions Extend Expiry.



Status - It will show the deployment status of build pipeline including all the components involved like tomcat, springboot, cassandra, kafka, storm and docker. User can also restart any services (tomcat/springboot) by selecting the checkbox present against the component .



View - User can view all the deployment related properties like graphical representation of pipeline build , whitelisting and components involved in the deployment, build name, namespace, status etc .

Details - It also same as view .

Delete **-** With delete feature user can delete the deployed pipeline from the sandbox env i.e K8.

Download - User can download values.yml file of deployed pipeline. It's the same file which Helm uses for deployment through concord in vajra native K8 cluster.

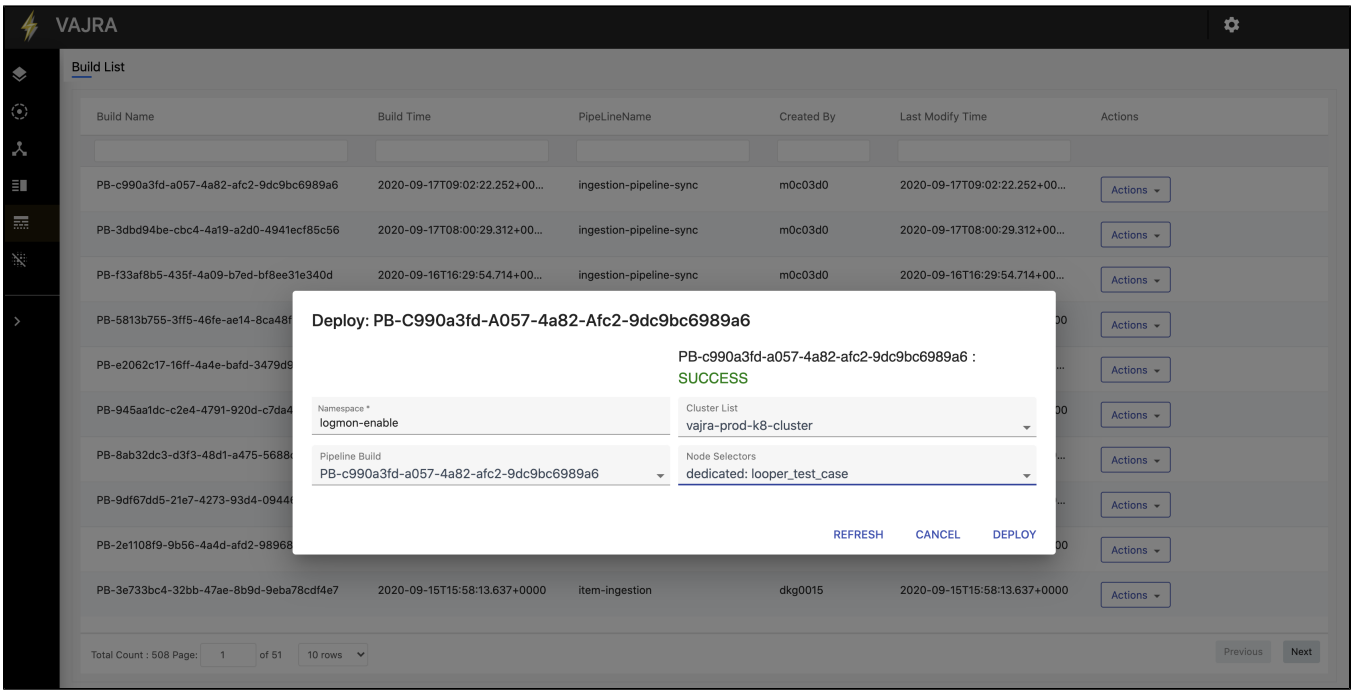
ConcordLog - User can also view the deployed pipeline concord log though this feature .

Deployment Upgrade

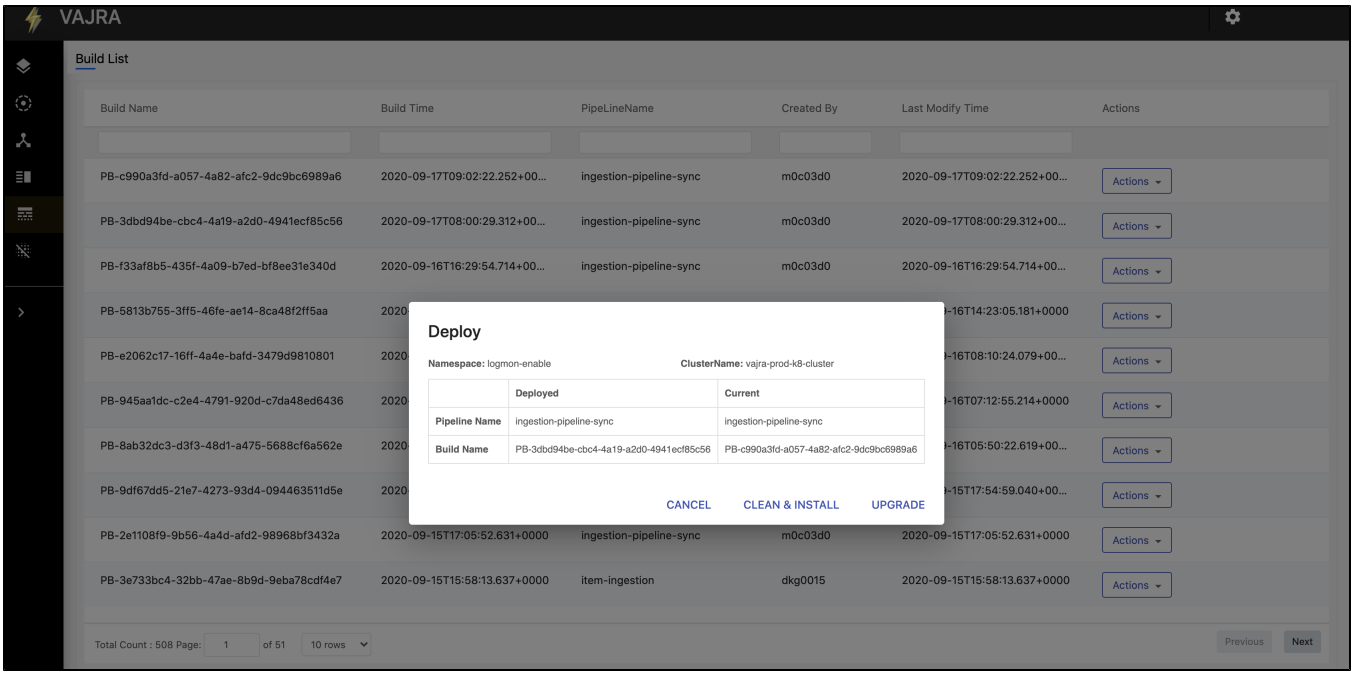
Usecase :- Suppose user has to change the component details like change in artifact version of an application involved in the pipeline design without re-deploying the whole pipeline .

Solution :- Change the component details like version and then do the build of pipeline to include latest component artifact into build and then click on **Deploy** option of new build from **Build List** page. Please note first two records of below screenshots, where `**PB-3dbd94be-cbc4-4a19-a2d0-4941ecf85c56**` was old build for `**ingestion-pipeline-sync**` pipeline and after version change of an application user has triggered new build for the `**ingestion- pipeline-sync**` from the **Pipeline List** page and got new build as - `**PB-c990a3fd-a057-4a82-afc2-9dc9bc6989a6**`.

Now user click on **DEPLOY** option from Actions drop down list of Build List page for new build mentioned above . Enter all the required details like Namespace, Cluster List, Node Selectors and click on DEPLOY .



After Clicking on **DEPLOY** option as shown above user will get two options again i.e **UPGRADE** which will just deploy the changed component in the existing pipeline and **CLEAN & INSTALL** which will first clean the existing pipeline and then deploy the whole pipeline with new build .



Steps to check the logs for deployed pipeline :-

TOMCAT | SPRINGBOOT LOGS:-

For checking the logs of tomcat, springboot you need to login into vajra cluster and then by kubectl log cmd you can check the application console logs Or if application is using LOGMON then enable it by configuring the proper datacenter to write to a logmon file.

STORM LOGS :-

1. First replace the below URL with your namespace where sync-ingestion-pipeline is namespace & prod.vajra.k8s.us.walmart.net is cluster DNS .

http://mergedstorm-122.sync-ingestion-pipeline.prod.vajra.k8s.us.walmart.net/index.html

2. From Topology Summary select any topology you want to check the logs .

3. From Worker Resources click on port link and it will give you below link
http://stormsupervisor122-1:8000/log?file=productPipeline_catdev-7-1600274475%2F6701%2Fworker.log

4. In the above link replace :8080 with namespace and cluster DNS like below and then you will get appropriate worker node logs.
http://stormsupervisor122-1.sync-ingestion-pipeline.prod.vajra.k8s.us.walmart.net/log?file=sourcePipeline_catdev-4-1600274468%2F6700%2Fworker.log

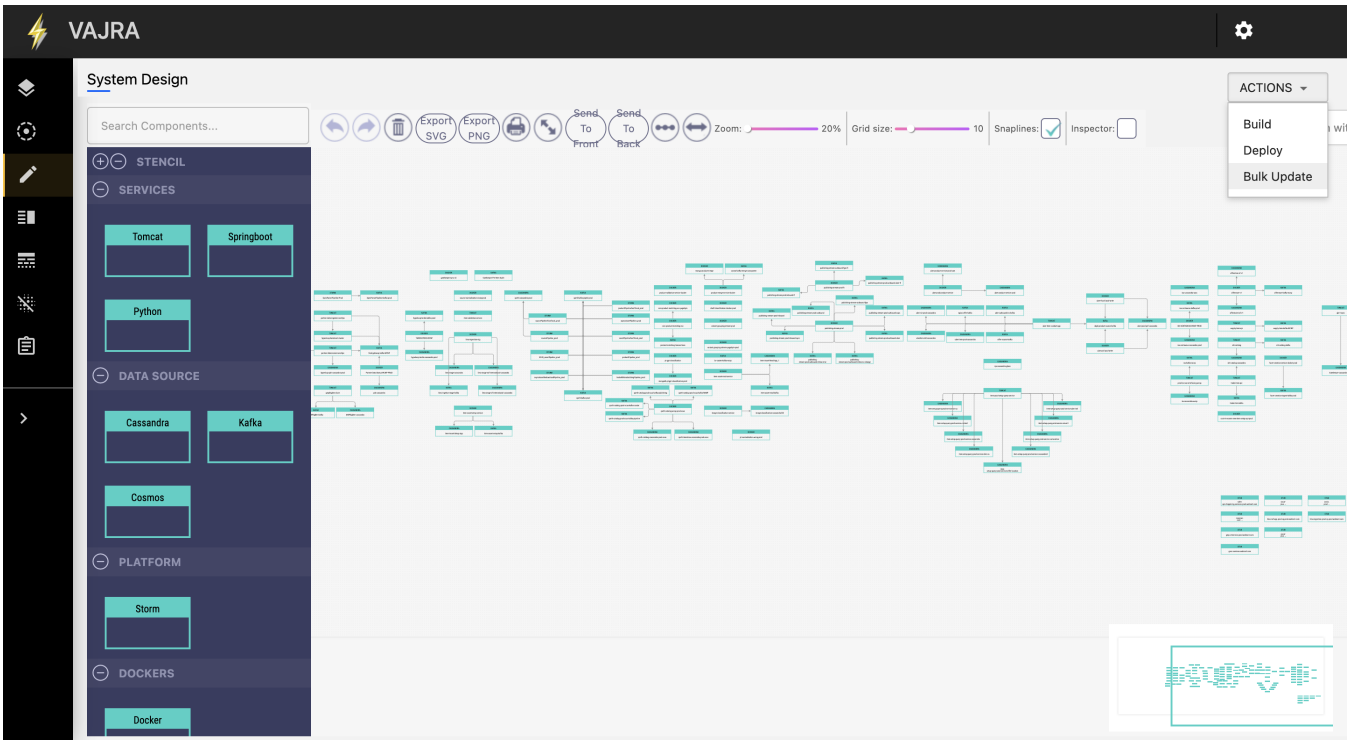
KAFKA | CASSANDRA LOGS:-

For datasource also first login into vajra cluster and then using kubectl log cmd you can view the container logs and also you can ssh into container through kubectl cmd .

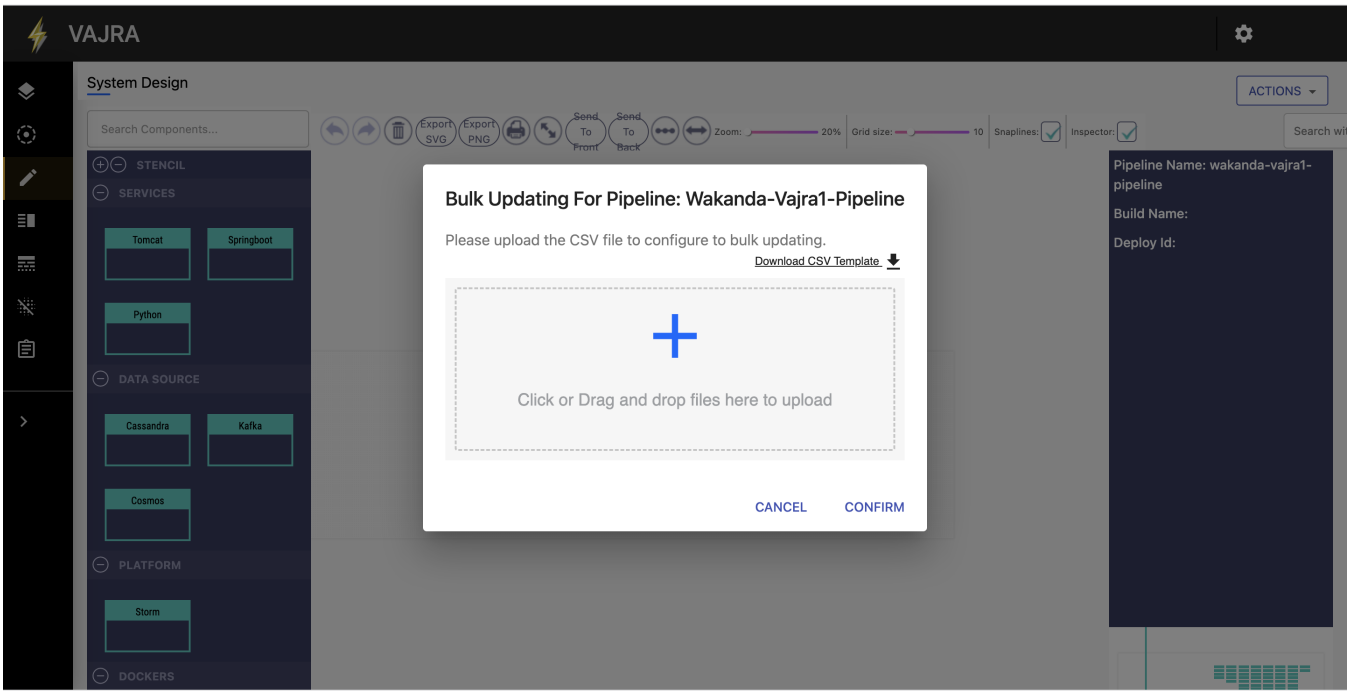
NOTE: Please find below the most common kubectl commands to view container logs and interact with the container - Kubectl Commands

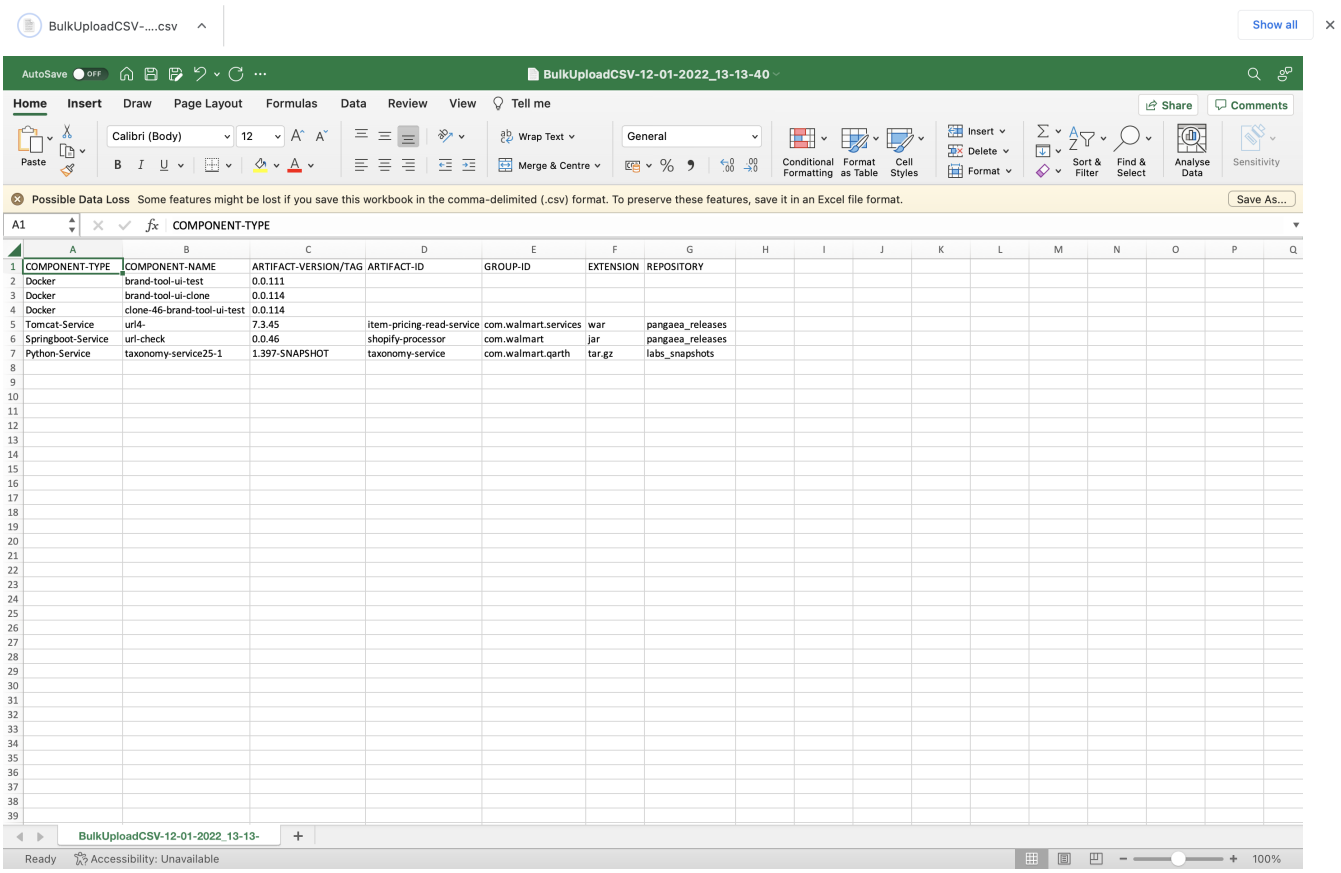
Component Bulk Update Based On Pipeline

In case users need to update the version/tag/artifact details of all the components(tomcat, spring boot, python, docker) of an existing pipeline in groups without actually updating them individually, the **Bulk Update** option available in pipeline design page (ActionsBulk Update) can be used.



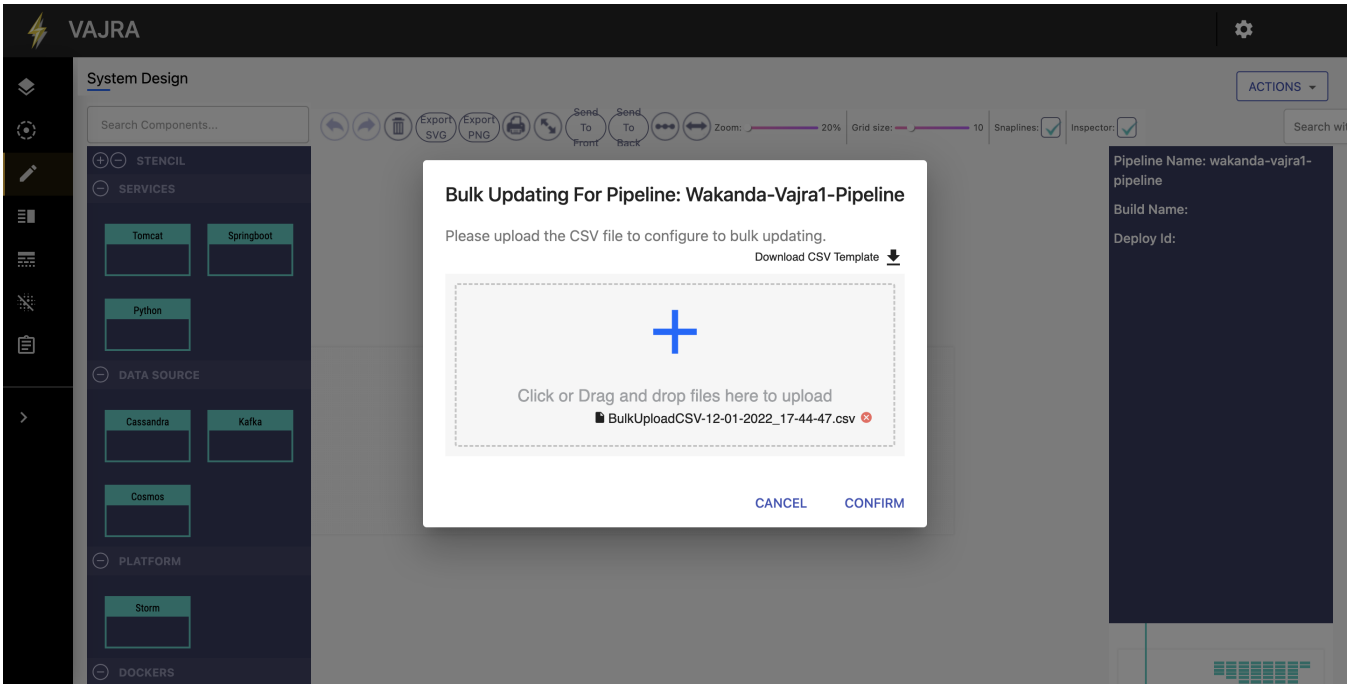
on clicking "Bulk Update" option, a dialog box appears where a csv file containing the details(component-type, component-name, artifact-version/tag, artifact-id, group-id, extension, repository) of all the components will be downloaded.





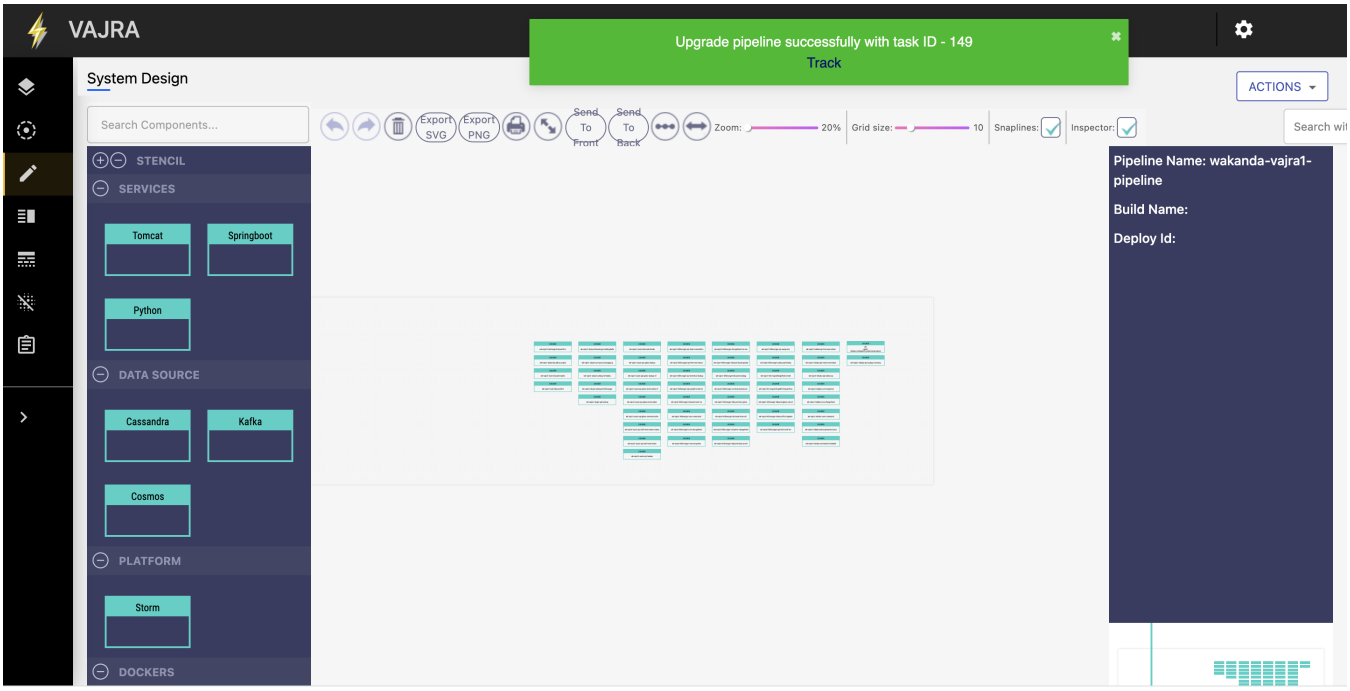
COMPONENT-TYPE	COMPONENT-NAME	ARTIFACT-VERSION/TAG	ARTIFACT-ID	GROUP-ID	EXTENSION	REPOSITORY
Docker	brand-tool-ui-test	0.0.111				
Docker	brand-tool-ui-clone	0.0.114				
Docker	clone-46-brand-tool-ui-test	0.0.114				
Tomcat-Service	url4-	7.3.45	item-pricing-read-service	com.walmart.services	war	pangaea_releases
Springboot-Service	url-check	0.0.46	shopify-processor	com.walmart	jar	pangaea_releases
Python-Service	taxonomy-service25-1	1.397-SNAPSHOT	taxonomy-service	com.walmart.qarth	tar.gz	labs_snapshots

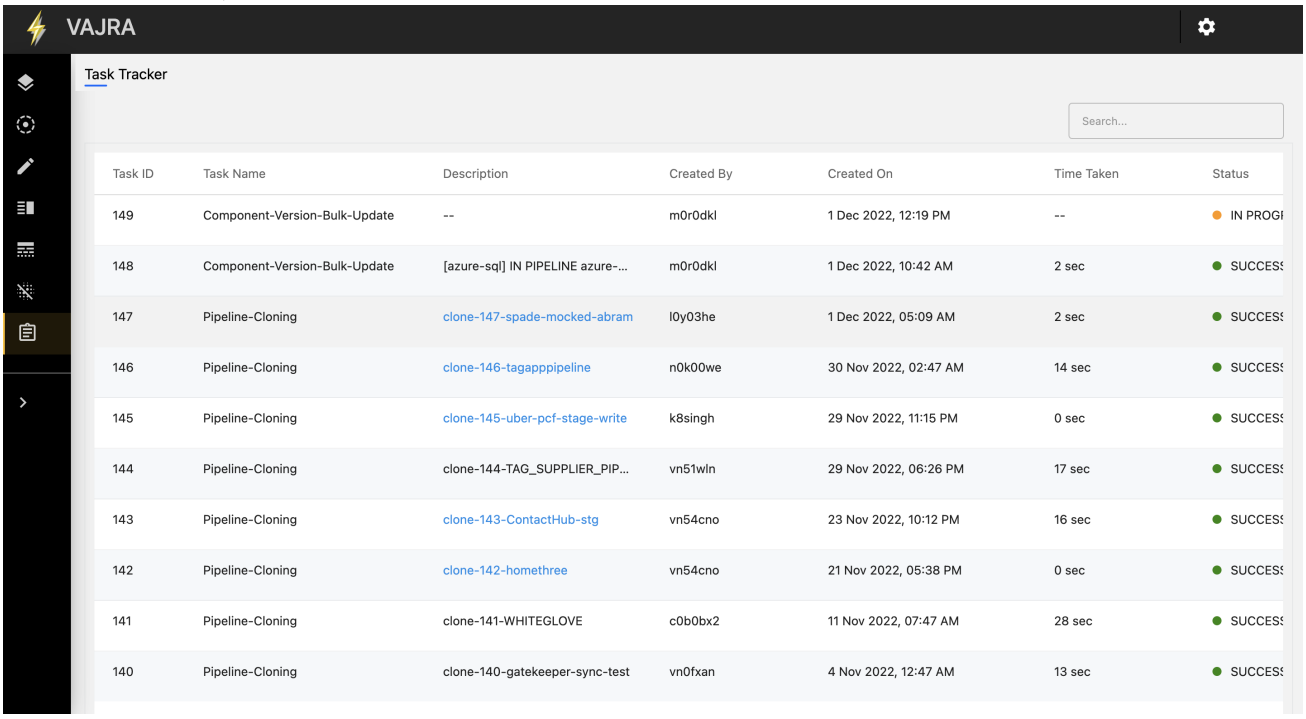
Now, the version details can be updated in that csv file referring to the component name and its type and the same should be uploaded back and click on confirm. In case any component need not be updated, either it can be left as it is or that particular row can be removed from the table (optional).



Once uploaded, a task will be created to track on this upload process, which can be checked from task tracker table. task details will be shown on ui upon submitting the file.

In the task tracker table, you can find if its success or failed.
In case success, the updated component names is displayed in the description column along with the pipeline name.
In case of failure, the components updated so far in the process(till before the component for which update failed) is displayed in the description.

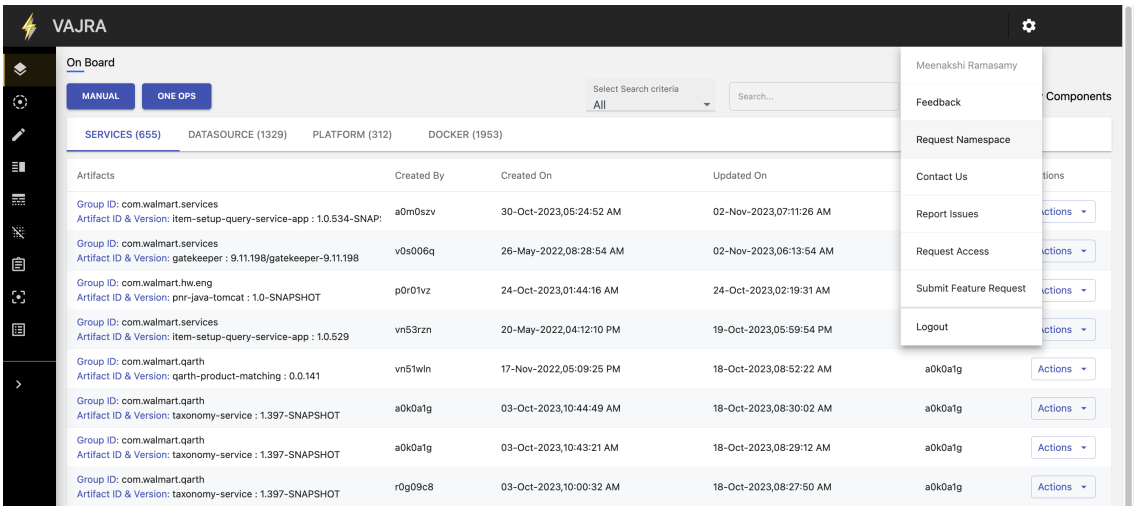




Namespace Request

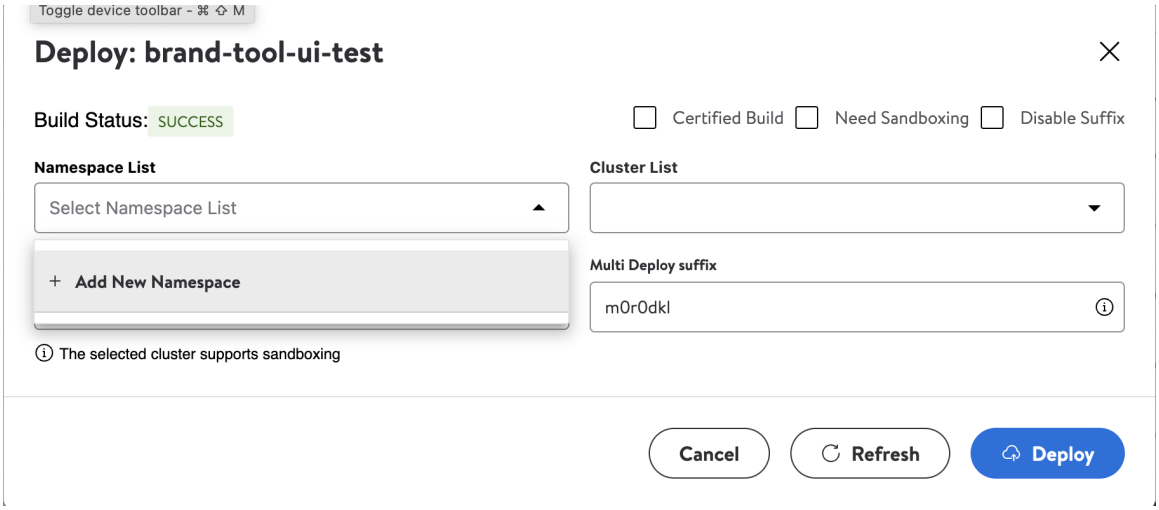
It is required to register your namespace details with vajra to proceed with the deployment. There are two options available
1.To register your existing wcnp namespace details with vajra using 'RequestNamespace Option' following the instruction below.

a. To request for a new namespace for deploying from vajra, Select settings options from onboarding page Request Namespace. Once submitted, kindly let vajra team / vajra onboarding channel know to approve the request.



2. Incase you dont have any wcnp namespaces available, for poc reasons, you can request Vajra team to provide namespace in vajra clusters. For this, you have to contact the vajra team.

3. Another option is to create a new namespace from dx console or the same can be done via vajra ui also using 'Add New Namespace' option available as shown below



Access Restriction

1. Only selected user can login into vajra application. Please contact #vajra-onboarding slack channel for login related queries.
2. User will be able to modify the component/pipeline which they own. Other users components/pipelines they can't modify.
3. User can view any other user's components for their pipeline design.
4. In case you need to modify already existing pipeline owned by others you can "Clone" in the pipeline listing screen from the "Actions" and modify the design as needed and save it as new pipeline.

Quota Requirements

Based on the pipeline design, number of pipelines and the number of users from your team wants to test and run the pipeline we can arrive at the compute requirement. This compute capacity cost will be borne by the team / cost centre. The compute quota has to be made available in the azure cloud for integrating with our k8s cluster to use it. The step by step guide to avail this compute quota will be made available soon. As of now please contact #vajra-onboarding channel for assistance.

Kubernetes Cluster Access

Please contact #vajra-onboarding channel to get read only access to the kubernetes cluster in which you have deployed the pipeline. We can use this to connect to docker container running within your namespace and tail the logs of the container if needed. This can be used along with the vajra-cli too if needed.

External TCP Option

When cassandra/kafka is deployed as part of the pipeline, it cannot be accessed externally i.e it is not exposed so that can be connected from any external sources.

Functionally, this isn't a problem. the pipeline flow works as expected. But incase, you want to read or write data in the dbs, have to sh into the respective db containers and using cqlsh and kafka commands it can be achieved.

But, dbs can be provided with an external dns similar to the services deployed in vajra. This is achieved through this external tcp option.

- This option is supported only with vajra single tenant clusters. Not supported by wcnp clusters.
- To use this option, contact vajr ateam via #vajra-onboarding channel requesting to enable external tcp option. Once provided, you can see a checkbox - 'External TCP' while deploying. check that and proceed with deployment. Upon deployment, you can get an external dns and port for the db as well.

Deployment Expiry

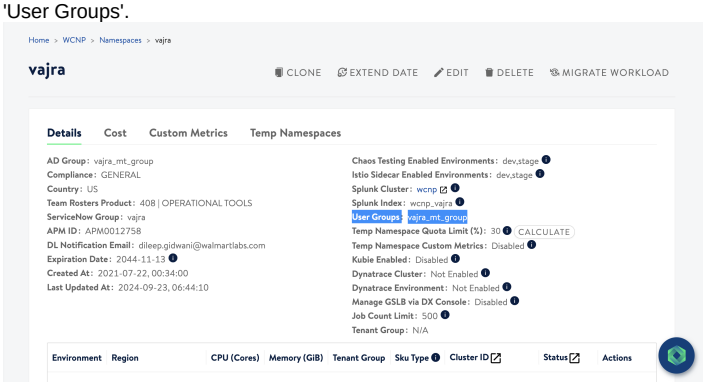
Expiry refers to the time by which the deployment is deleted automatically. The expiry date and time can be viewed in the deployment page against your deployment under 'Build Expiry Date' column.

Vajra Clusters (wus-dev-aks-vajra) :

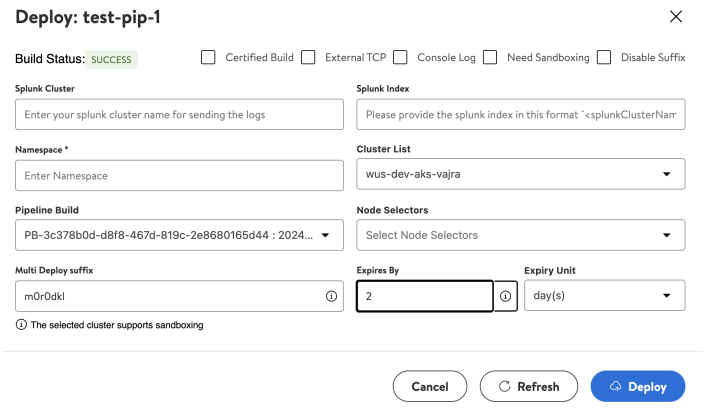
- When deployed in vajra clusters, the default expiry is 1 day. So, after 1 day from the time of deployment, it is deleted.

WCNP Cluster :

- To delete resources deployed in wcnp clusters, Vajra need to have access to the namespace used for deployment. This can be checked from dx console dashboard (as shown in the reference image attached below) with respect to the namespace if vajra ad Group ('***vajra_grp***') is part of 'User Groups'.



- If the mentioned ad group 'vajra_grp' is not part of it, please add it.
- Once given permission for vajra to your namespace, please let the vajra team know with namespace details for which it has been set.
- Now, once its added users wcnp deployments can also be deleted automatically as it is set.
- Setting the expiry, there are 3 ways to do so (deploying from UI)
    1. *During Deployment:* you get the option to set it in deployment window while trying to deploy. This is set for that particular deployment only.
    2. *Pipeline Level*: It can also be set at pipeline level. To do so, edit Pipeline click next you can can see Deployment Expiry option. This applies to this pipeline whenever deployed. This can be overridden using above method while deploying.
    3. *Namespace level*: Setting up expiry at namespace level will delete any deployment being deployed in this namespace.



Note: Priority is set as 1, 2, 3. Whatever is et during deployment is taken. Otherwise pipeline level set value (if any) will be taken. Otherwise namespace level set value (if any) is taken. Finally if nothing is set, default value 1 day is set.

- To set expiry in plugin, just set it as n(d/h). n number of days/hours. d days. hhours. To use this as well, for wcnp clusters vajra ad group should be part of User groups in for the specified namespace.

MPS Kafka Connectors:

Teams using mps kafka connectors in their applications can make use of this mps docker component available with Vajra.

Follow these steps to onboard mps into your pipelines

1. Clone the docker component - 'mps-vajra'.
2. With the cloned one, Under Env Variable section modify these values accordingly
    a. TOPICS - the topic from which mps will be consuming messages. Only one topic can be configured.
    b. GROUP_ID - give any name
    c. HTTP_API_ENDPOINT - the http endpoint to which mps should push the message to.
    d. WORKER_BOOTSTRAP_SERVERS - kafka end point where the topic exists. incase of vajra deployed kafka, give the external dns and port details (kafka in vajra has to be deployed with external tcp option enabled). For actual kafka, broker details can be given.
    e. Don't change/modify/remove the other env variables available in the cloned component.
3. Under Env Variables section, you will find three configs named => WORKER_CONFIGS_TOPIC, WORKER_OFFSET_STORAGE_TOPIC, WORKER_STATUS_STORAGE_TOPIC having values compName-config, compName-offset and compName-status respectively. Rename them. it has to be unique for every mps component.
4. Once all the above steps are done, include the component in your pipeline and deploy them.
5. Once deployed and mps logs are loaded, you need to start it using the following end point.
    a. http://<mps dns you get after deploying in vajra>/connectors/start_all
6. Now, you are ready to push the message in your kafka topic mentioned in mps configs which will be posted to the end point mentioned via mps. you can check logs to make sure you have received messages.

NOTE: Make sure to disable your application from connecting directly with kafka.

For Support

#vajra-onboarding slack channel any assistance.