**Title: Creating a CI/CD Pipeline**

## **Introduction**

Continuous Integration and Continuous Deployment (CI/CD) is a development practice that enhances software delivery by automating integration, testing, and deployment. This document outlines the steps to create a CI/CD pipeline using industry best practices.

## **Prerequisites**

* A version control system (Azure Git repo).
* A CI/CD tool (Azure DevOps).
* Service Connection for Kubernetes Cluster.
* Self-Hosted agent.
* Variable Group: - "We have used a Variable Group to securely store the username and password for pulling images from the Azure Container Registry (ACR)."
* Helm charts.

### **Version Control System Setup (Azure Git Repo)**

#### **1. Repository Creation**

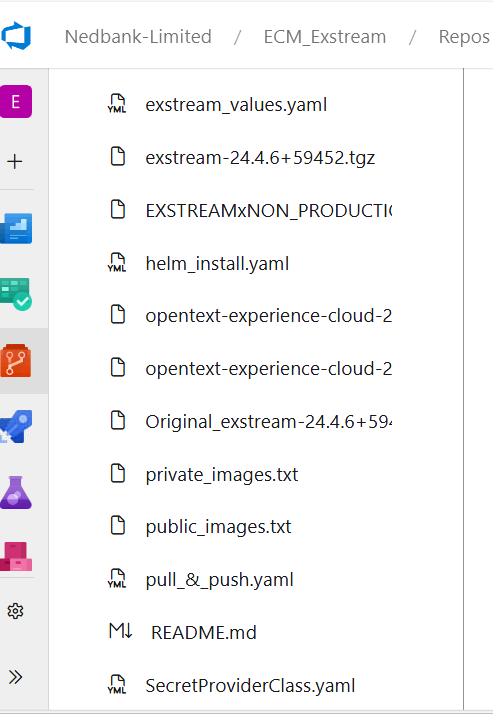
A version control system using **Azure Git Repo** has been set up to manage source code efficiently. A dedicated repository, **ECM\_comp\_Exstream\_dev**, has been created to facilitate streamlined development and collaboration. This repository will serve as the central source of truth for all changes, ensuring versioning, tracking, and secure storage of code and configuration files.



#### **2. Purpose of the Repository**

The **ECM\_comp\_Exstream\_dev** repository is used for:

* **Code Versioning**: Keeping track of changes and maintaining the modification history.
* **Collaboration**: Allowing multiple developers to work together.
* **Pipeline Automation**: Storing CI/CD YAML configurations for automated deployments.
* **To store the Application code.**



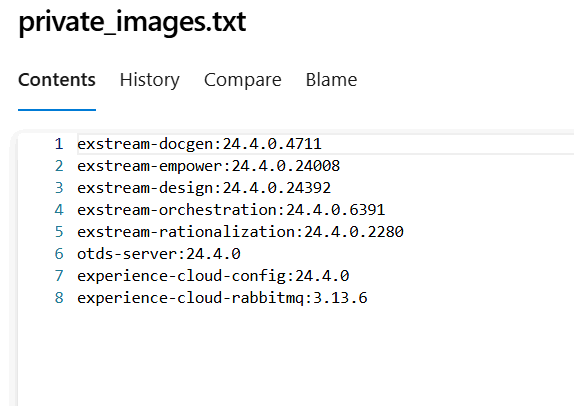
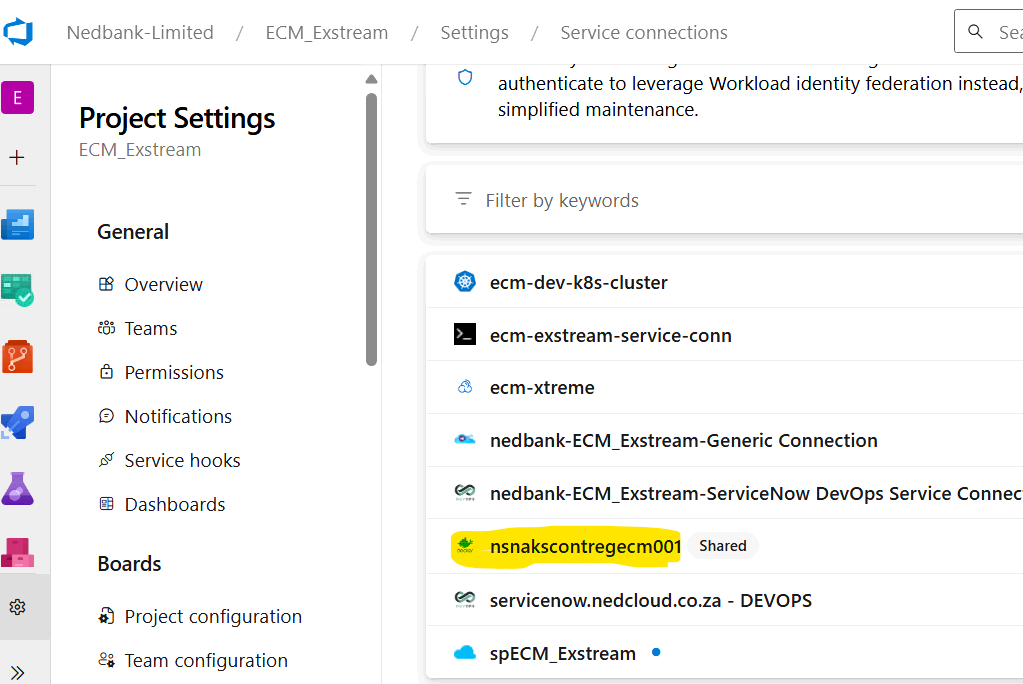
#### **3. YAML File Configuration for Public and Private Images**

As per the requirement, YAML files have been added to handle both **Public and Private** container images. These YAML files play a crucial role in automating the CI/CD pipeline, making deployments more efficient.

##### **a. Handling Public Images**

* The pipeline pulls public container images from official repositories (such as Docker Hub, Microsoft Container Registry).
* No authentication is required for pulling these images.
* The YAML configuration ensures that the latest available versions are used automatically.
* Below are the screen shot for public images.
* 

##### **b. Handling Private Images**

* The pipeline is configured to pull **private images** securely from **Azure Container Registry (ACR)**.
* A **Variable Group** has been used to securely store the **username and password**, ensuring that sensitive credentials are not exposed.
* The YAML file contains steps for authentication with ACR, pulling the required images, and deploying them.
* Below are the screen shot for Private images.
* 
* **4. Security Measures**
* **Azure DevOps Service Connections** are used to manage access to ACR securely.
* **Secrets Management** is implemented using **Azure Key Vault** or **Variable Groups** to avoid direct exposure of credentials in the pipeline.
* 
* **Installation and Configuration of OTDS and RabbitMQ**

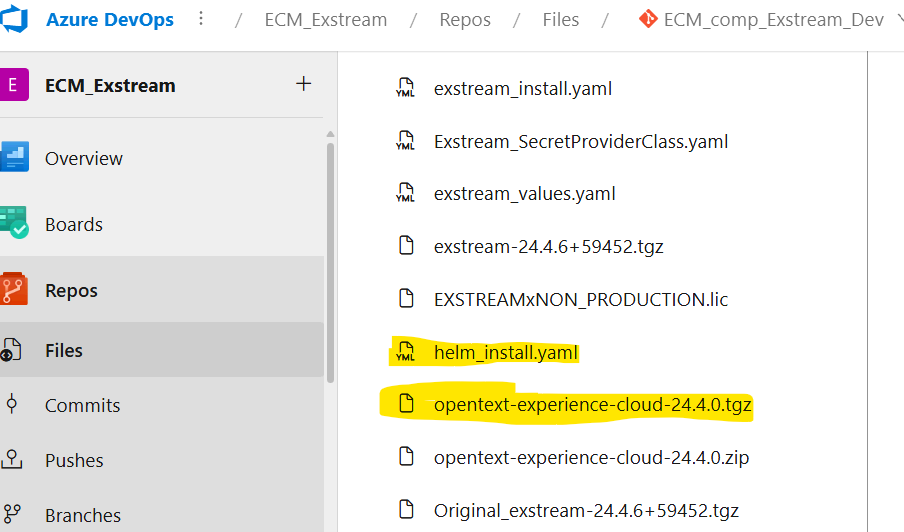
#### **1. Installation of OTDS and RabbitMQ**

* As per the project requirements, **OTDS (OpenText Directory Services)** and **RabbitMQ** have been successfully installed. These components play a crucial role in authentication, authorisation, and message queuing within the application.

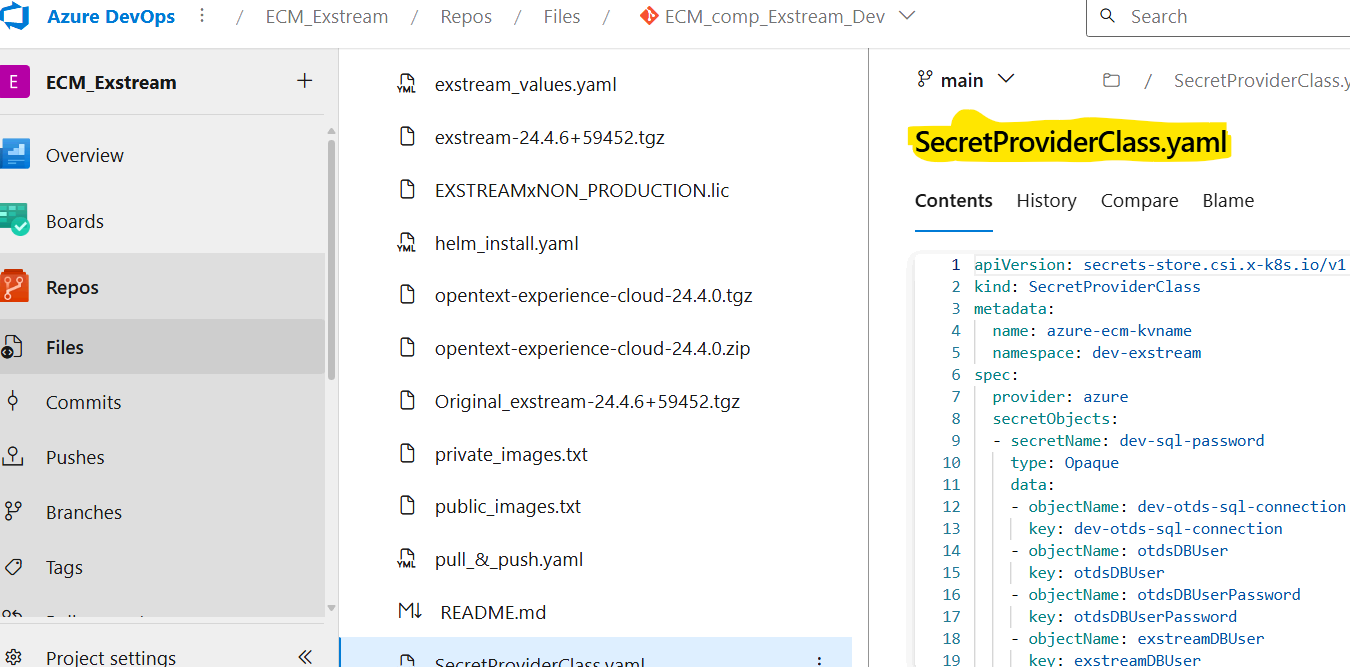
#### **2. Tgz File Provided by Client**

* The client has provided a **Tgz (Tarball Gzip) file** containing necessary configurations and setup files.
* This file was extracted to retrieve the required resources and configuration settings.

#### **3. Addition of Values YAML File**

* After extracting the **Tgz file**, the necessary values were identified and configured.
* A YAML file was created based on these values and added to the **Azure Git Repository**.
* The file was renamed as **experience\_values.yaml** to align with the project's naming conventions.
* Additionally, a **Helm\_install.yaml** file was added to automate the deployment process using Helm in the CI/CD pipeline.
* **4. Pipeline Integration**
* The **Helm\_install.yaml** file is used to trigger the deployment of OTDS and RabbitMQ through the CI/CD pipeline.
* The pipeline ensures smooth and automated deployment with minimal manual intervention.
* 
* **Secure Configuration and File Handling in the Repository**

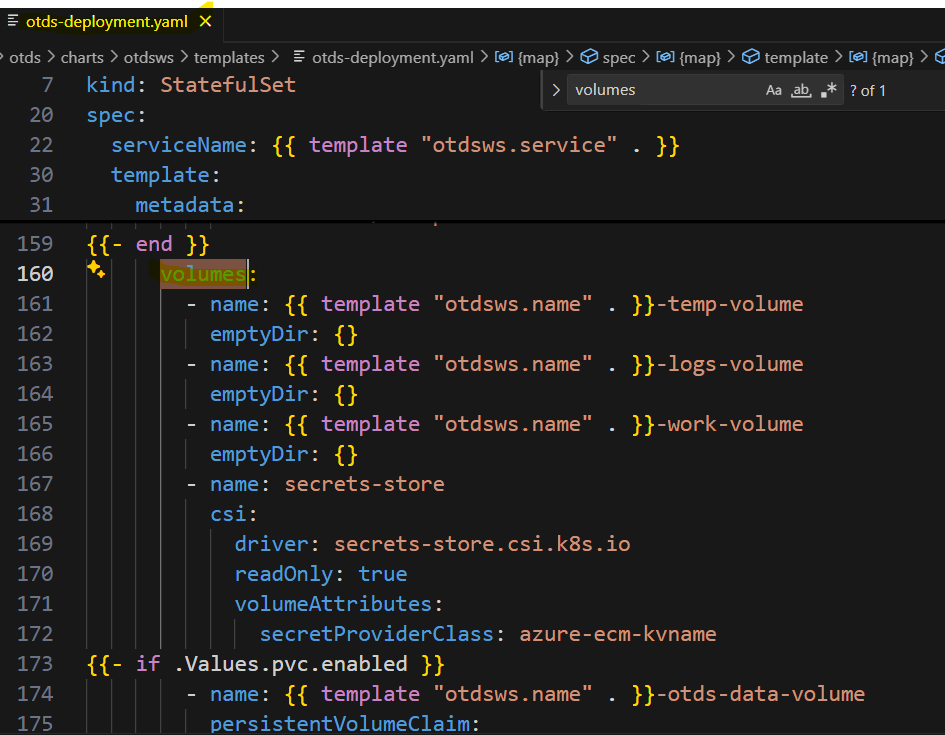
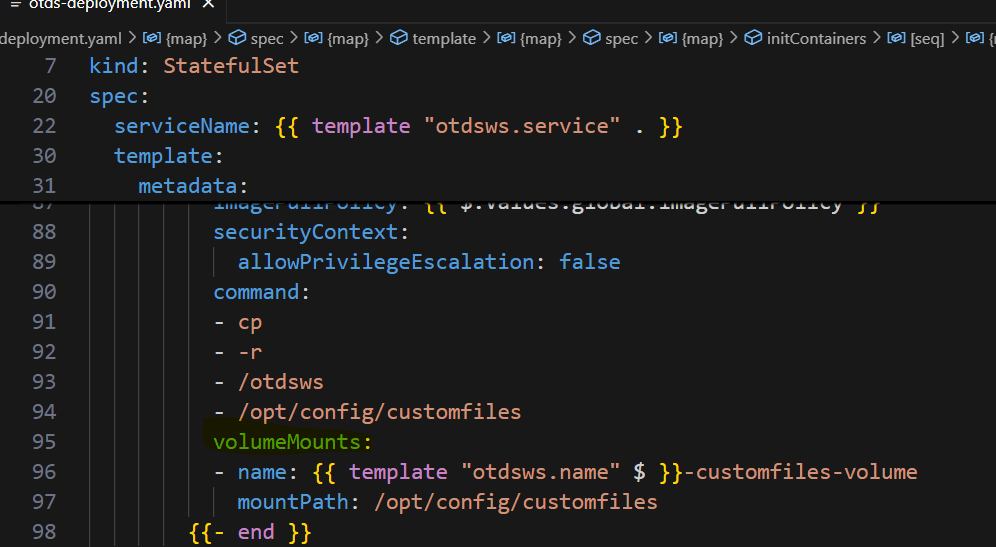
#### **1. Password and Secrets Management**

* Passwords and secrets are securely fetched from **Azure Key Vault (KV)**.
* A **class provider YAML file** has been added to the repository for managing access.
* 

#### **2. Azure Key Vault Implementation**

* **Azure Key Vault** is used to securely manage and store keys, secrets, and certificates. It ensures that sensitive data such as:
* Connection strings  
   are stored securely and accessed only by authorised services. This prevents credential leaks and enhances security.

#### **3. Handling of Tzg Files**

* The **Tzg file** provided by the client has been extracted.
* **Deployment Configuration Update**
* After extracting the .tgz file for **OpenText Experience Cloud 24.4.0**, the otds-deployment.yaml file located at the path  
   charts/otds/charts/otdsws/templates/ was updated as per the project requirements.
* This file is part of the Helm chart structure and is responsible for defining the StatefulSet deployment for OTDS Web Services (otdsws).  
   The changes include updates in volume definitions, metadata templates, and service bindings using Helm templating.
* As per the requirement, **Volume Mounts and Volumes** were configured with the correct **username and password**.
* The updated files were then converted back to a **Tzg file**.
* 
* The newly generated **Tzg file** has been uploaded to the repository for further use.
* 
* 
* **Update to experience\_values.yaml File**

#### **1. Modification as per Requirement**

* The **experience\_values.yaml** file has been updated to meet the project requirements. The key modifications include:
* **Namespace Addition**: A specific namespace has been added to ensure proper isolation and organisation of resources.
* **JDBC Connection String Update**: The JDBC string has been updated with the necessary database details to enable secure connectivity.

#### **2. JDBC Connection String Details**

* The updated **JDBC connection string** now includes:
* **Database Host**: The SQL Server instance.
* **Port**: 1433 (default SQL Server port).
* **Database Name**: ECM\_OTDS.
* **Security Parameters**: Encryption and server certificate validation enabled.
* **Login Timeout**: Set to 30 seconds to ensure optimal connection performance.

#### **3. Repository Update**

* The updated **experience\_values.yaml** file has been committed to the repository to maintain version control and facilitate automated deployment via the CI/CD pipeline.
* **Update to experience\_values.yaml File – Service Account Configuration**

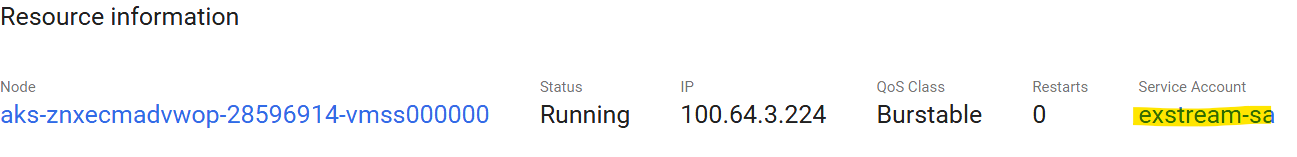
#### **1. Addition of Service Account**

* The **experience\_values.yaml** file has been updated to include a **Service Account** with the name **exstream-sa**. This service account ensures secure authentication and access control for Kubernetes workloads.

#### **2. Purpose of Service Account**

* Provides a dedicated identity for applications running in Kubernetes.
* Enables secure communication with other services and APIs.
* Helps in managing role-based access control (RBAC) effectively.

#### **3. Repository Update**

* The updated **experience\_values.yaml** file has been committed to the repository, ensuring that the new service account configuration is included in the deployment process.
* 

### **Deployment Status – CI/CD Pipeline Execution**

#### **1. Pipeline Execution Success**

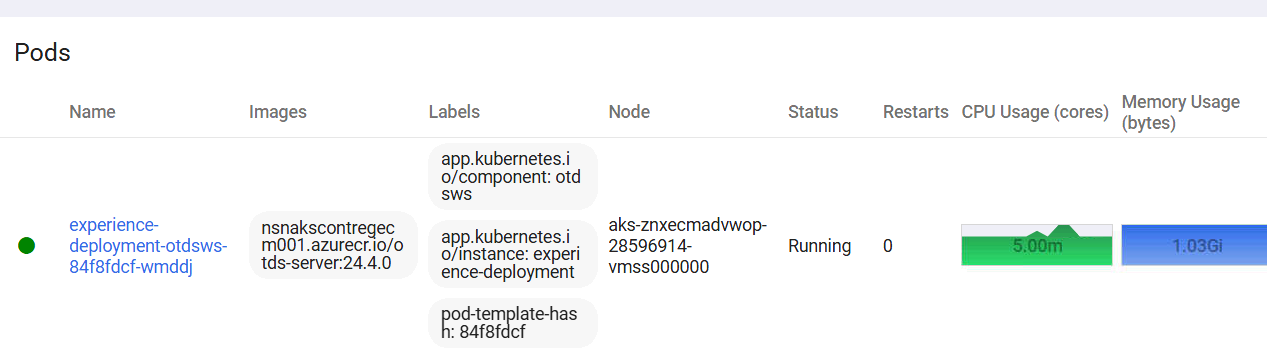
The CI/CD pipeline has been successfully executed, ensuring smooth deployment of the required components.

#### **2. Deployment of OTDS and RabbitMQ**

* The **OTDS (OpenText Directory Services)** and **RabbitMQ** pods have been deployed successfully.
* Both services are now **running and operational** without any errors.
* The deployment logs confirm that all configurations and dependencies were correctly applied.

#### **3. Verification and Status Check**

* The status of the deployed pods has been verified.
* All services are in a **healthy state**, indicating a successful deployment process.

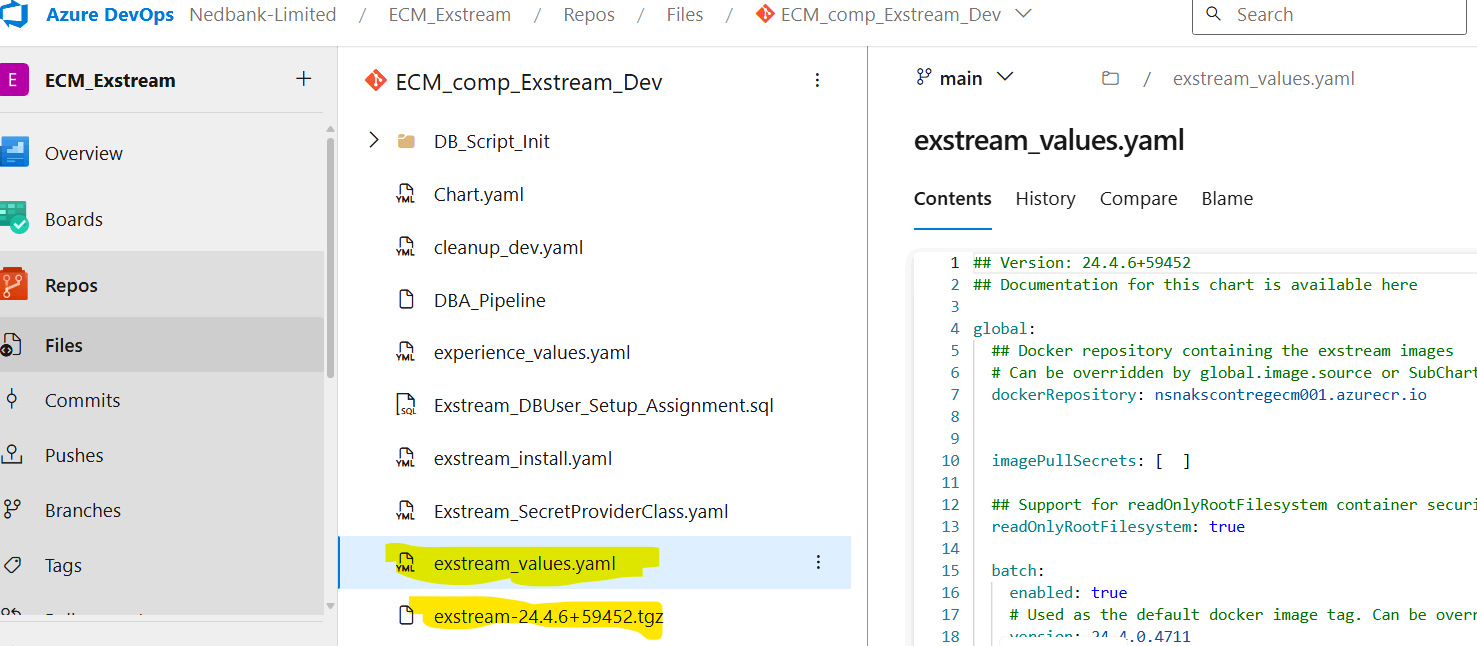


### **Handling of Exstream Tgz File – Repository Update**

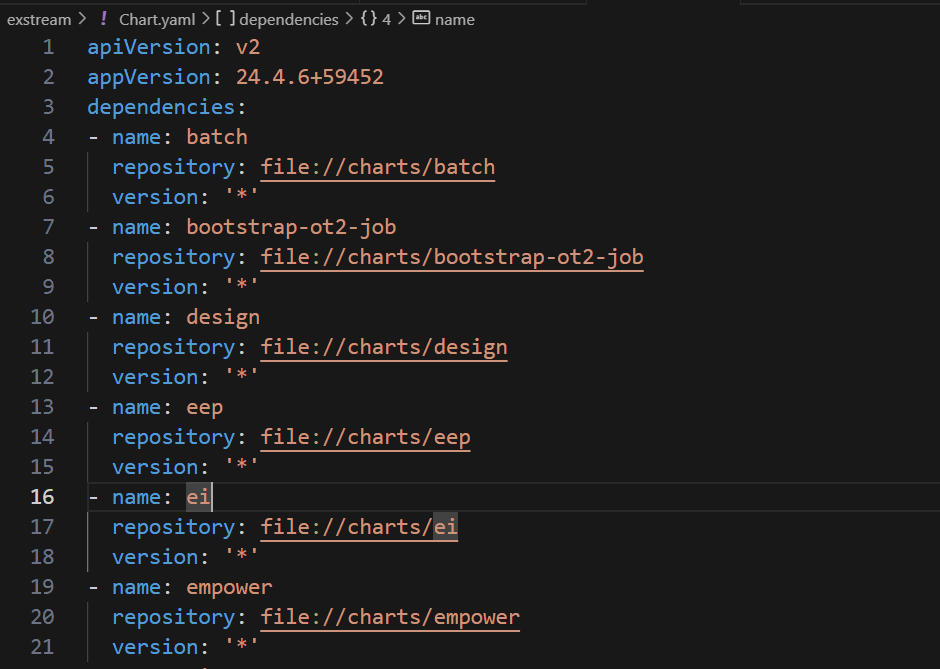
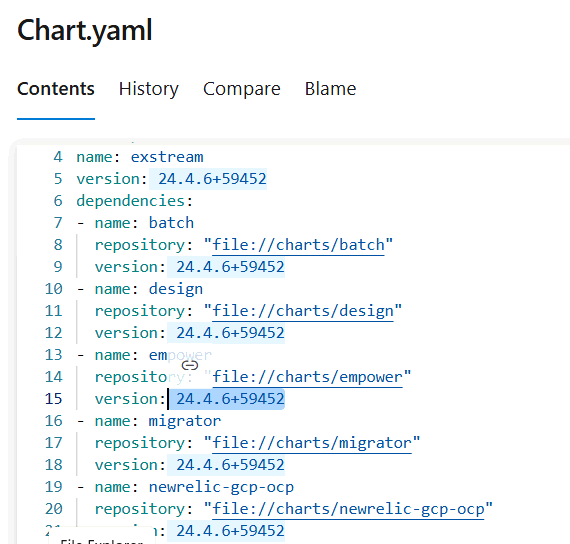
#### **1. Extraction of Exstream Tgz File**

A new **Exstream Tgz file** was provided as part of the deployment process. This file has been successfully extracted to access its contents.

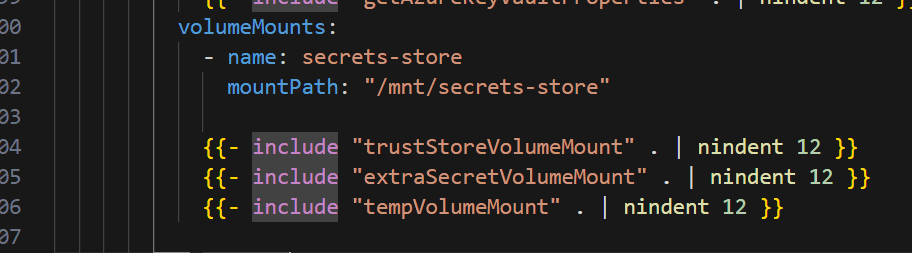
#### **2. Addition of Exstream\_value.yaml File**

* After extracting the Tgz file, the necessary configurations were identified.
* A new YAML file, **Exstream\_value.yaml**, was created with the required values.
* This file was then added to the repository to ensure proper configuration management.
* 
* **Update to Exstream Tgz File – Values and Volume Mounts Configuration**

#### **1. Extraction of Exstream Tgz File**

* The **Exstream Tgz file** has been successfully extracted to access its contents and apply the necessary configurations.
* **Modification of Chart.yaml**
* The Chart.yaml file has been updated to include **new images and tags** as per the latest requirements.
* This ensures that the latest container images are referenced for deployment.
* 
* **Addition of Chart.yaml in Repository**
* The updated Chart.yaml file has been added to the repository to maintain version control and ensure seamless integration with the CI/CD pipeline.
* The repository now reflects the latest dependencies and configurations.
* 

#### **2. Addition of Values and Volume Mounts**

* The required **values** and **volume mounts** have been configured for the files listed in the screenshot.
* 
* These configurations ensure proper resource allocation and seamless functionality of the following components:

##### **Design**

* design.yaml
* post-upgrade-job.yaml
* design-schema-upgrade-job.yaml

##### **Orchestration**

* orchestration.yaml
* orchestration-janitor-cron.yaml
* orchestration-cron-job-init.yaml

##### **Empower**

* empower.yaml
* empower-schema-upgrade-job.yaml
* empower-editor-upload-job.yaml

##### **Rationalization: -**

* Rationalization.yaml
* Rationalization-schema-upgrade-job.yaml
* Rationalization-post-upgrade-job.yaml
* The updated files were then converted back to a **Tzg file**.
* 
* The newly generated **Tzg file** has been uploaded to the repository for further use.

### **Schema Details and JDBC Configuration for Exstream**

#### **1. Schema Structure in Exstream**

Exstream consists of multiple schemas, each serving a specific function:

* **Design**
  + Used for **batch processing** and **on-demand execution**.
* **Orchestrator**
  + Handles **workflow orchestration** and **job scheduling**.
* **Empower**
  + Includes:
    - **Shared Schema**: Commonly used across various modules.
    - **System Schema**: Stores system-level configurations.
* **Rationalization**
  + Manages **optimization and data rationalization**.

#### **2. JDBC Configuration**

The database connection is established using the JDBC string:



* <DB\_HOST>: The database server hostname.
* <DB\_NAME>: The specific schema name to be connected.
* encrypt=true: Ensures secure communication.
* trustServerCertificate=true: Allows self-signed certificates for authentication.

#### **3. Implementation in Exstream**

* The JDBC connection string has been **added and configured** in the necessary YAML files.
* Each schema is **mapped correctly** to ensure proper database interactions.
* The updated configurations have been **committed to the repository** for version control.
* **Deployment Status After Successful Pipeline Execution**

#### **1. Pipeline Execution Summary**

* The pipeline **ran successfully** without errors.
* Upon completion, the **necessary pods** were deployed and started running.

#### **2. Pod Status After Deployment**

* The following components were successfully brought up after the pipeline execution:
* **Design**
  + Responsible for **batch processing and on-demand executions**.
  + Status: **Running**
* **Orchestrator**
  + Manages **workflow scheduling and execution**.
  + Status: **Running**
* **Empower**
  + Contains **Shared Schema and System Schema** for various functionalities.
  + Status: **Running**
* **Rationalization**
  + Handles **data optimization and processing tasks**.
  + Status: **Running**

#### **3. Verification Steps**

* Checked the pod statuses in **Kubernetes Dashboard**.
* Ensured that all services are **functioning correctly**.
* Resource utilization (CPU & Memory) is **within expected limits**.
* 