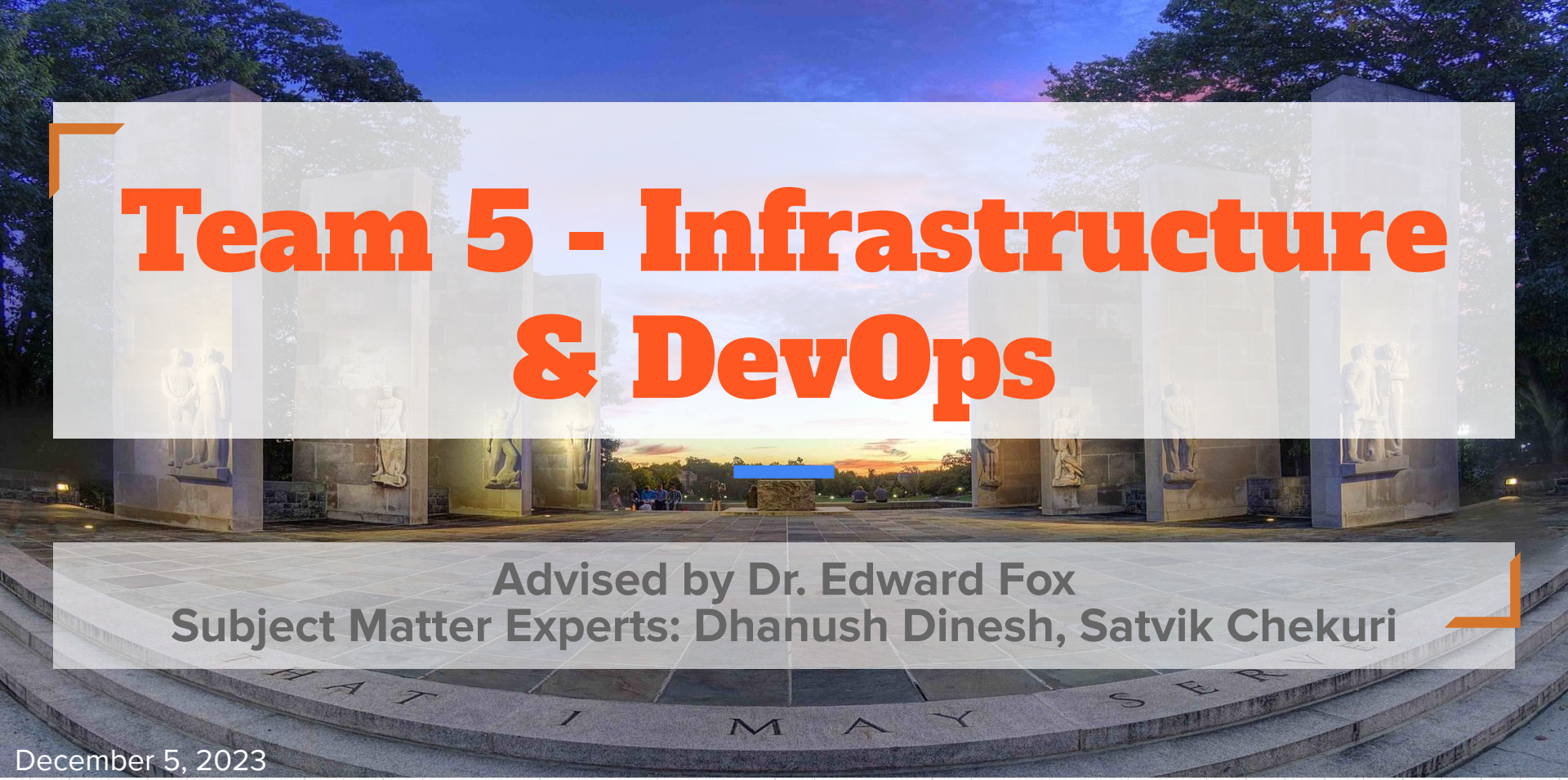


Team 5 - Infrastructure & DevOps

Advised by Dr. Edward Fox
Subject Matter Experts: Dhanush Dinesh, Satvik Chekuri

December 5, 2023

CS5604: Information Storage and Retrieval
Blacksburg, VA 24061 USA

VIRGINIA TECH™

Introduction

The Infrastructure and DevOps team is responsible for facilitating API enablement, providing Database and Filesystem Access via APIs, orchestrating CI/CD pipelines and ensuring smooth deployment to the Endeavour Kubernetes Cluster for the other teams.

Team Members:

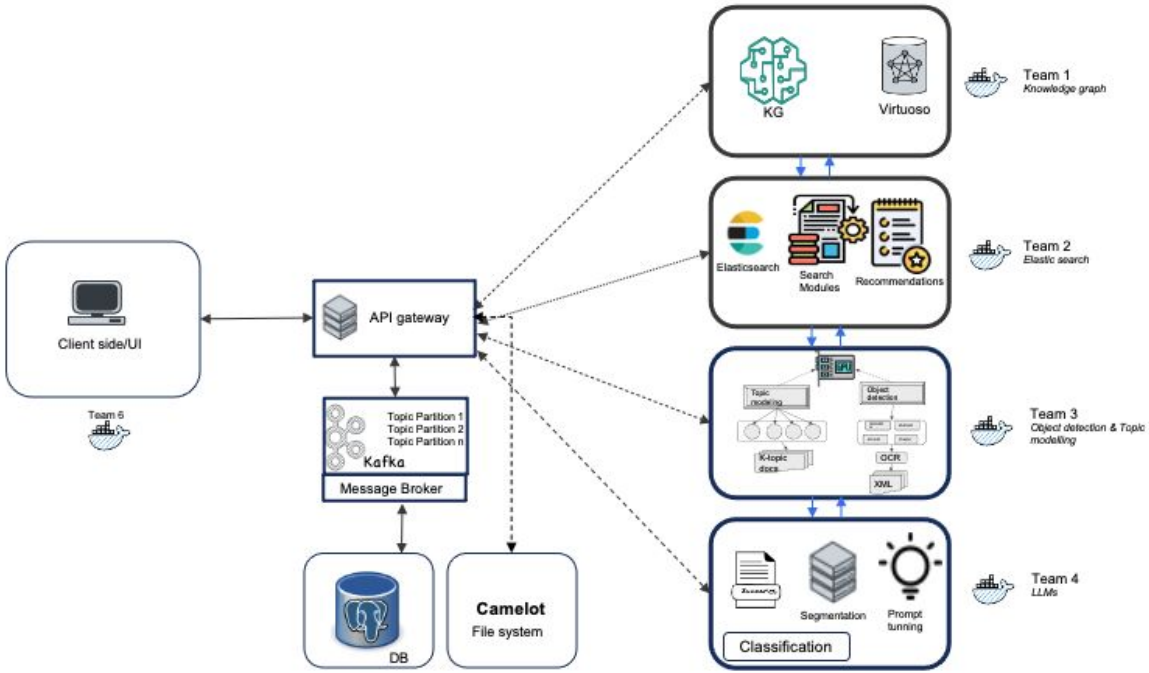
- Adeyemi Aina
- Amritha Subramanian
- Hung-Wei Hsu
- Shalini Rama
- Vasundhara Gowrishankar
- Yu-Chung Cheng

Agenda

- System Architecture
- Troubleshooting
- Kubernetes
- Kafka Usage and Monitoring
- APIs
- Database
- File System
- CI/CD
- Future Works

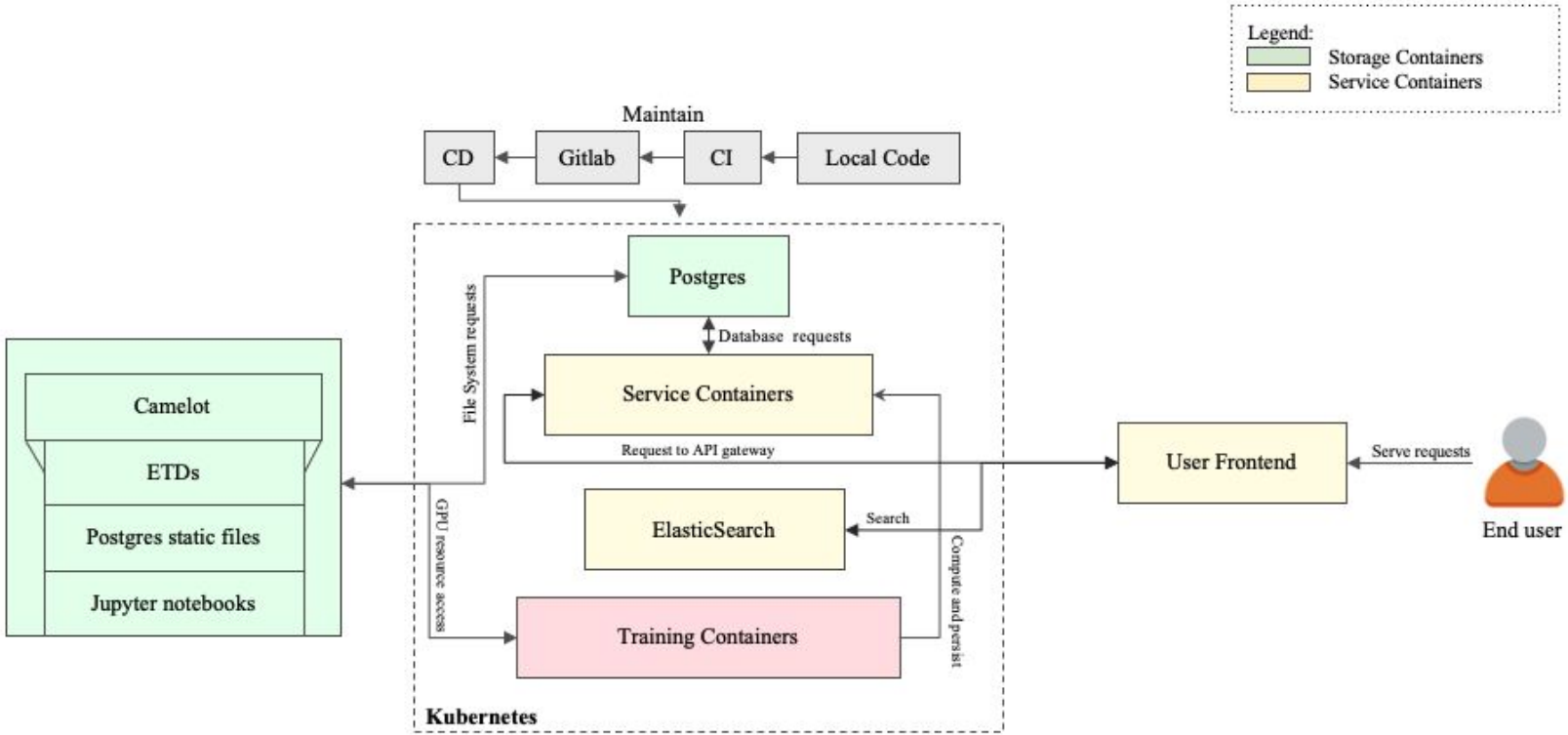
System Architecture

The design of the overall system is a Microservice architecture; each team's service is a suite of independently deployable modular service all communicating through an API gateway to other services. The diagram below shows a holistic view of the system architecture.



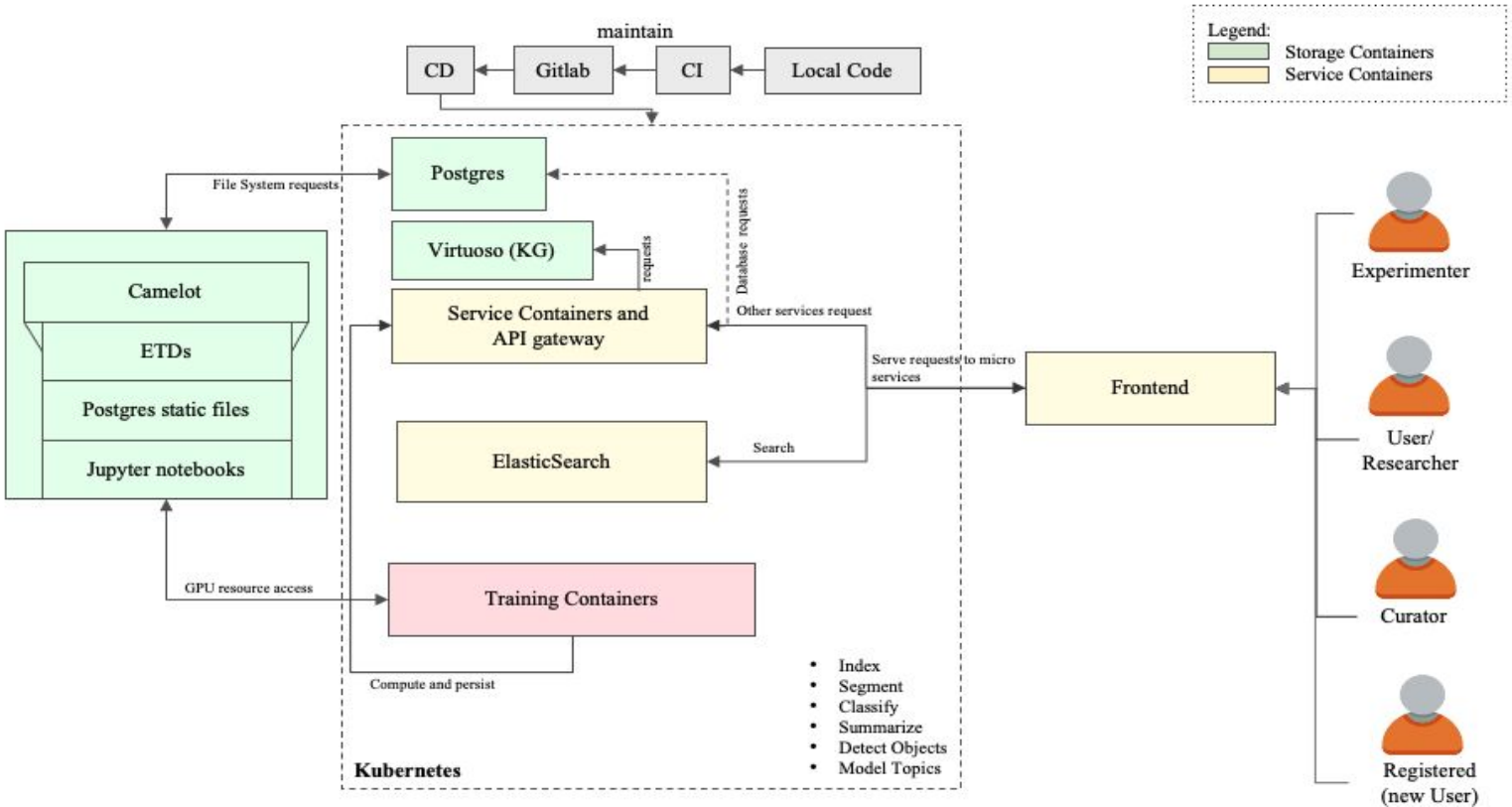
System Process Flow 1/2

The diagram below gives an holistic view of the system and process flow of the microservices, including a detailed understanding of how individual components behave and interact with each other.



System Process Flow 2/2

The diagram below gives a more detailed view of the system and shows the different personas interacting with the microservices from the other teams.

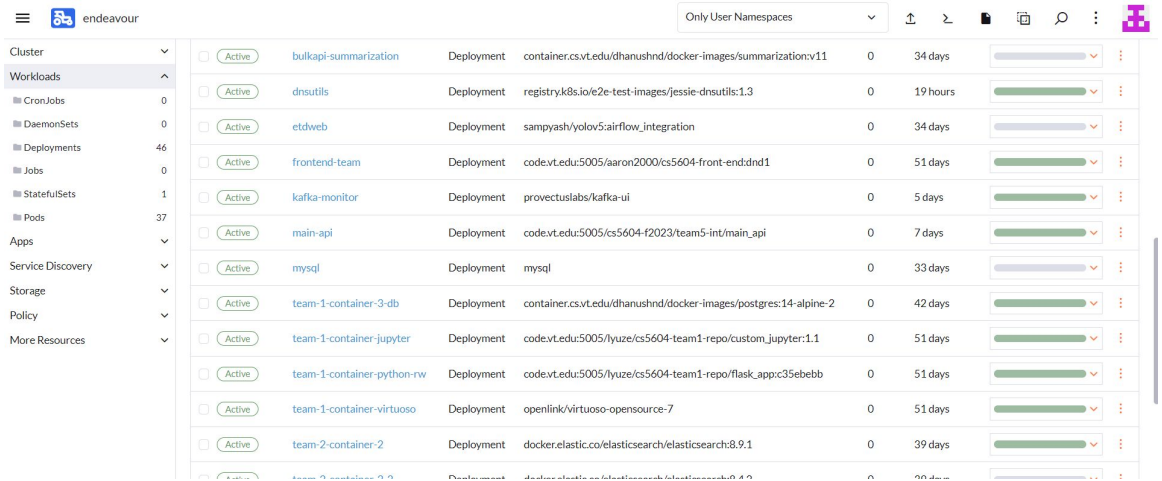


Troubleshooting

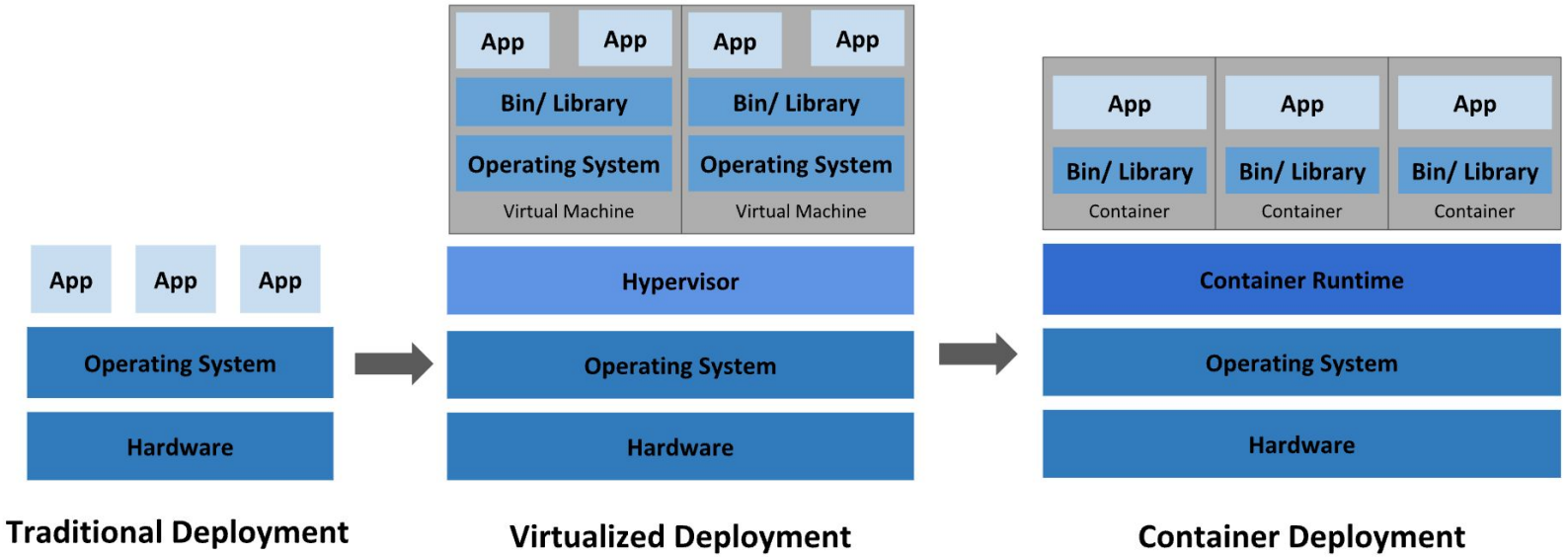
- **Security**
  - Provide a persistent storage to save the secure data.
    - Token
    - ElasticSearch config
- **CI/CD**
  - Disk
    - GitLab-runner will not clean Docker data.
    - Use cron job to clean it.
  - VM’s Maximum Transmission Unit (MTU) problem
    - Docker on VM can’t access GitLab.
    - Align the MTU for Docker with VM.

Migration

- **Provisioned cloud cluster resources:**
  - Created containers for each team with Jupyter environment.
  - Use Camelot to provision persistent file system to containers.
  - Use Ingress to enable external access to the cluster.
  - Migrate containers to Endeavour.



Kubernetes



https://kubernetes.io/images/docs/Container_Evolution.svg

Kubernetes & Rancher

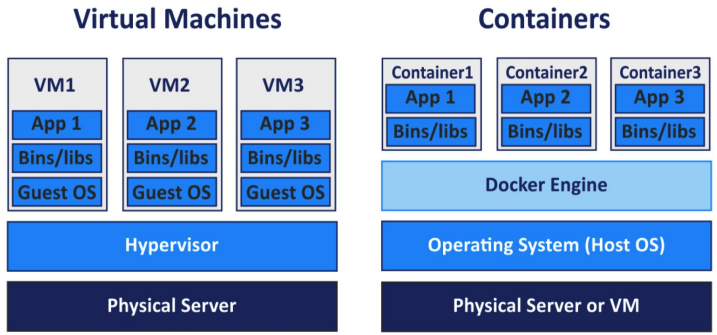
- **Kubernetes (k8s)**
  - Open source container orchestration tool.
  - Helps manage containerized application in different deployment environments.
  - 1 Master node, 1-N server nodes.

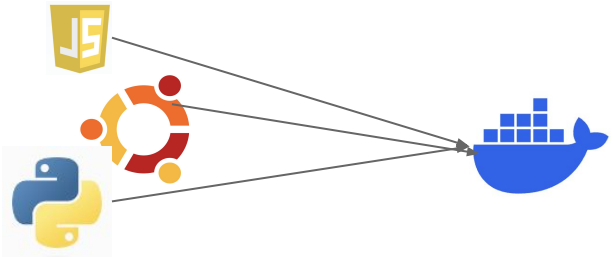
- **Kubectl** is the k8s command line tool to control the k8s cluster manager.

- **Rancher**
  - Open-source platform on top of k8s.
  - Provides all-in-one administration of clusters, easy-deployment.
  - Provides user interface for users to interact with the cluster; easy to get started.

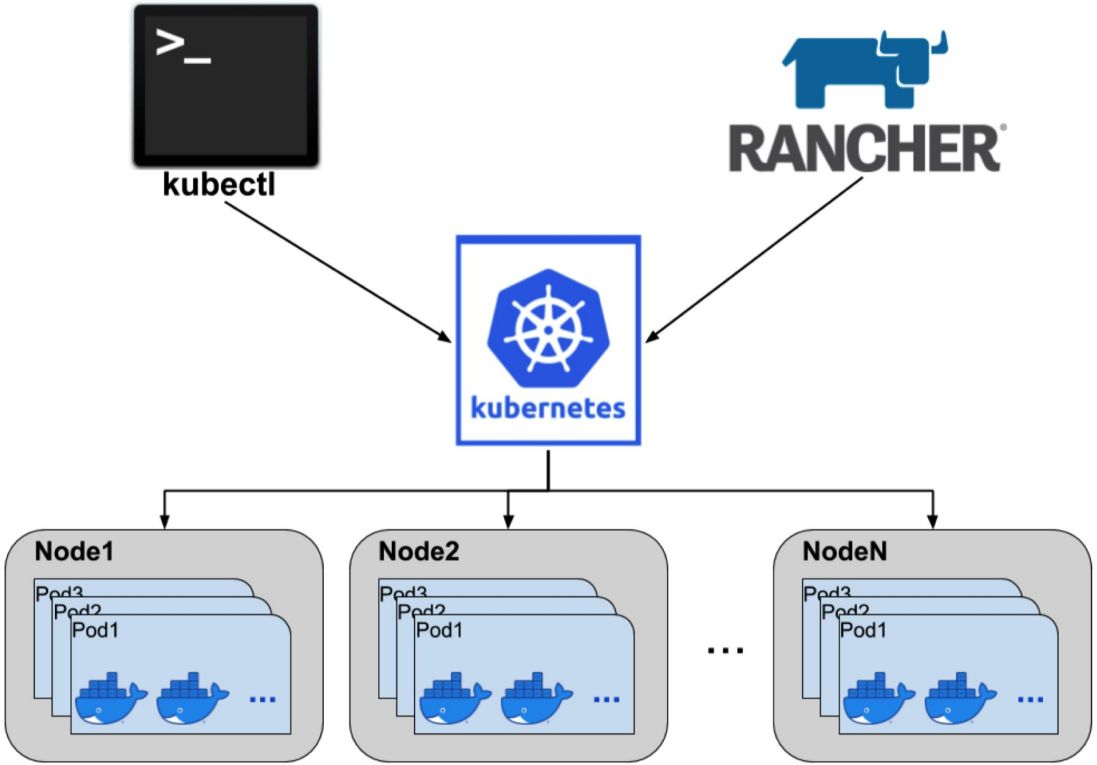
Docker and Containers

- **Docker**
  - Virtualization software
  - Easy Applications Deployment
  - Packages App with essential configuration and tools.
- **Container**
  - A standardized unit that includes everything needed to run an application.





Relationship between k8s, kubectl and Rancher



K8s Network

- Calico Crushing
  - Kubernetes' "service" can't forward network packets between each service.
- Reproduce
  - We built up a simple HTTP server to prove this issue to CS Tech Team.
- Resolve
  - CS Tech Team restart Calico to resolve it.

Kafka

- The example code is located at https://code.vt.edu/cs5604-f2023/kafka_example.
- How to create topic?
  - Just subscribe or produce a message with any topic you like.
  - You can check the topics on https://kafka-monitor.endeavour.cs.vt.edu/ui/clusters/broker-kafka/all-topics?hideInternal=true&page=1.
- What needs to be decided between each TEAM?
  - The topic name
  - The data format
    - String
    - Binary data
    - etc.
- Notice: the message should not be bigger than **1MB**.

API Implementation

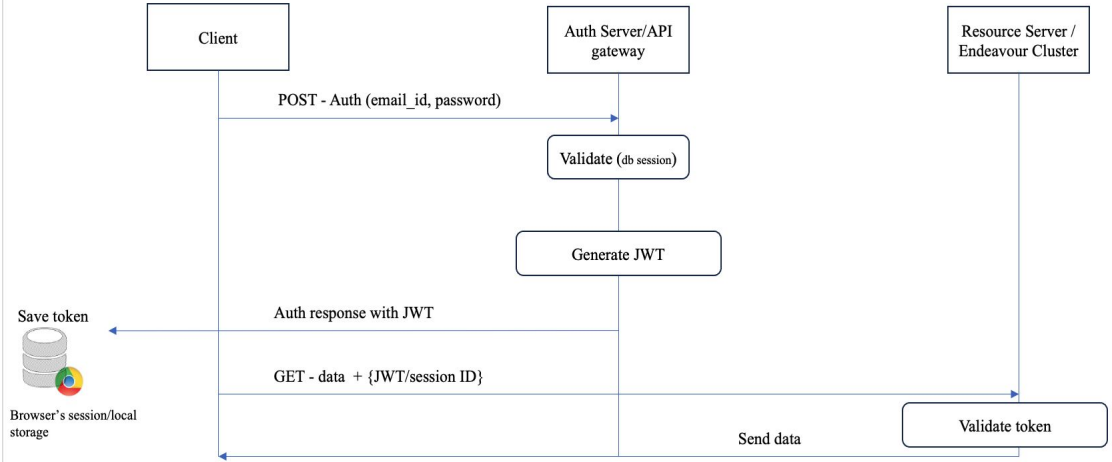
In the development of the APIs, we adhered to some key RESTful principles to ensure it was efficient, scalable, and easy to maintain:

- **Client-Server Architecture**: The client and the server operate independently, allowing each to evolve separately.

- **Uniform Interface**: A consistent and standardized approach to how the client interacts with the application. This includes using standard HTTP methods (GET, POST, PUT) and URLs to manipulate resources.

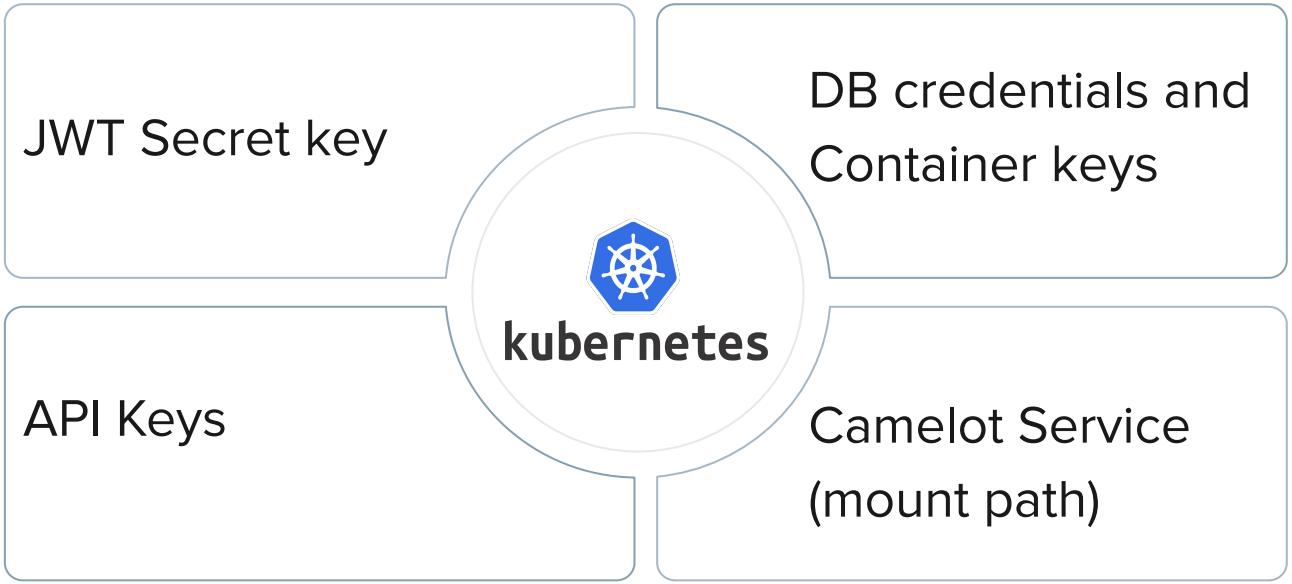
JWT Authentication

In the API gateway, a token-based authentication using JSON Web Tokens (JWTs) is implemented for user authentication. JWTs are an open standard that defines a compact and self-contained way for securely transmitting information between parties as a JSON object; the diagram below shows the cycle of the authentication.



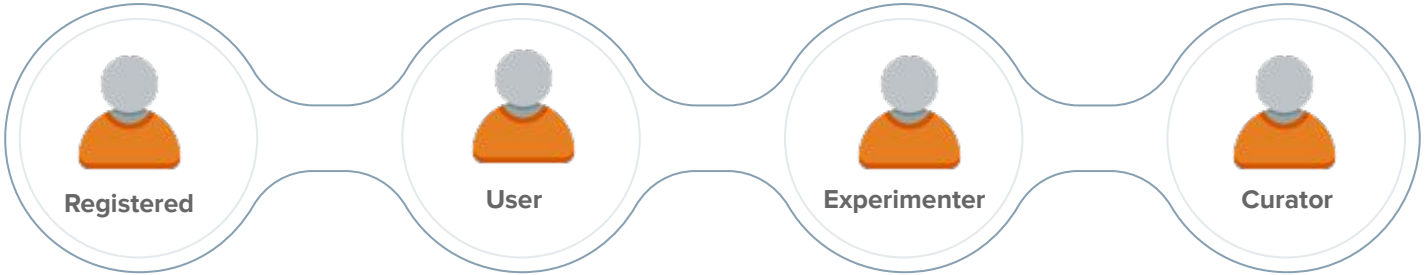
Security Measures

Sensitive Data Management: We used Kubernetes Secrets and config maps to store sensitive information for the Databases; Filesystem mount and other sensitive credentials are encoded on the Kubernetes cluster:



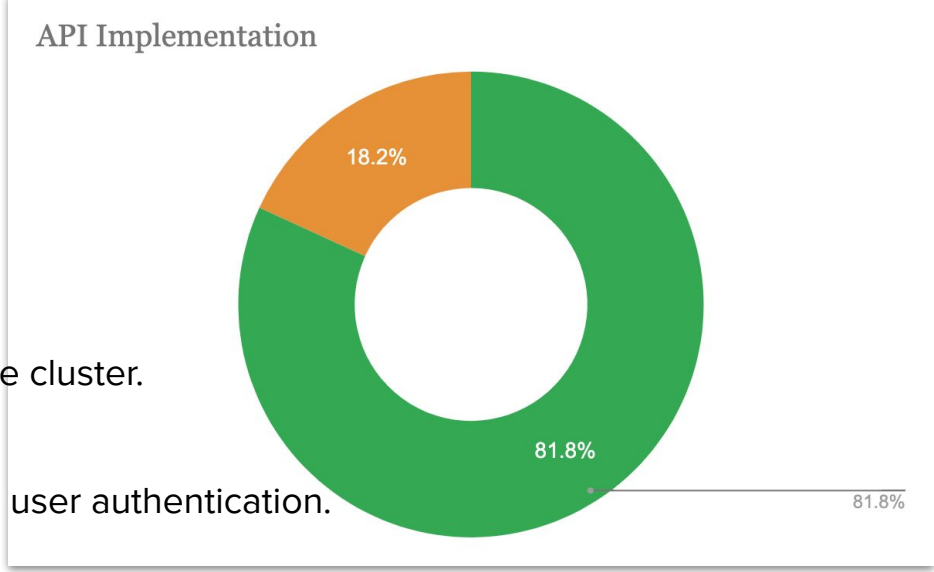
Role Based Access Control

- RBAC is a method of regulating access to a system or network resources based on the roles of individual users.
- **Roles Defined**: Curator, Experimenter, Registered, and User. Each role has specific permissions and access levels.



- **Access Control**: The system checks a user's role before granting access to different parts of the application. For example, only Curators have access to user management, Curator screens and all other features of the system.

APIs



- **Work done**
  - Completed API documentation.
  - Completed API gateway scaffold setup on the cluster.
  - Exposed the API for Database connection
  - Implemented Role Based Access Control.
  - Secured all endpoints with API keys, JWT for user authentication.
- **Progress**
  - Implemented 45 out of 55 APIs.
  - Address issues with the current implemented APIs.
  - Validating API list with the respective teams.
  - Write test routines for all API endpoints.
- **Future work**
  - Collaborate with the ElasticSearch team for a fine grained logging.
  - Begin implementation openapi standards documentation with swagger.

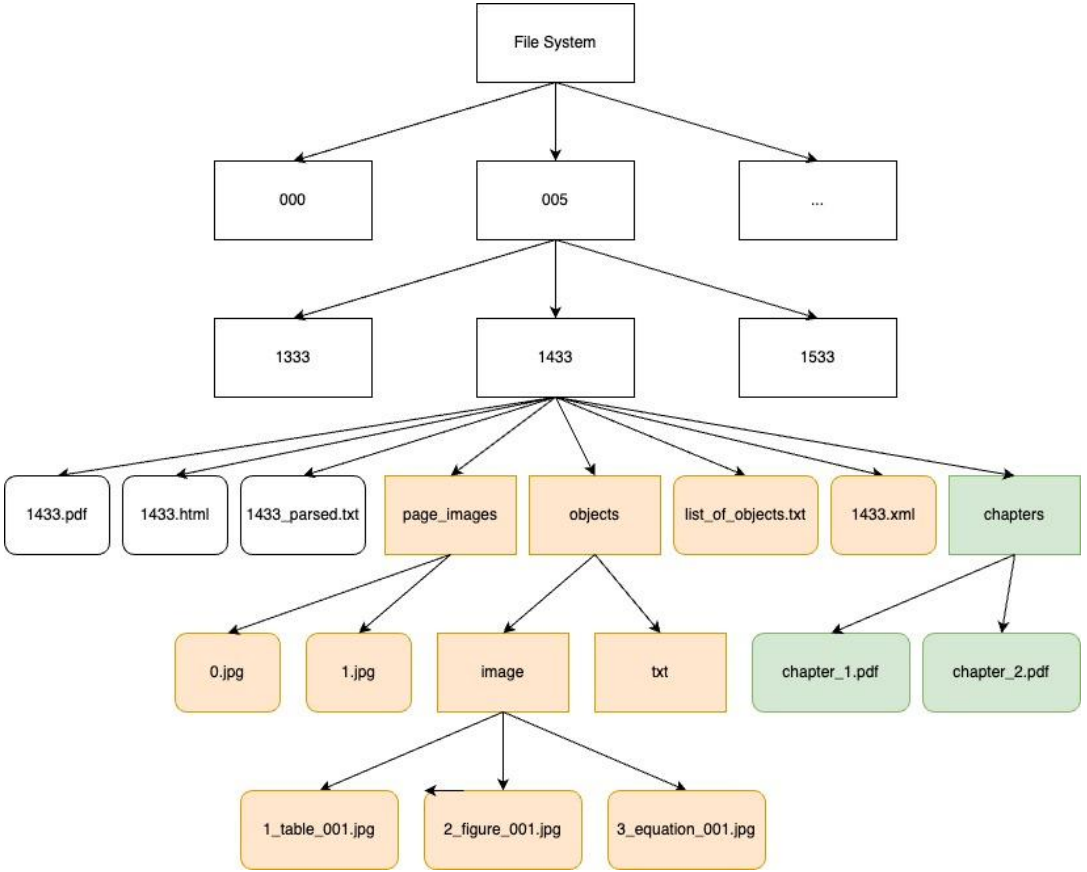
File system

- **Work done**
  - Added path information to new Database schema.
  - Created new path for 200+ segmented ETDs, update to Database schema.
  - Finalized the structure of the file system while considering the new Database tables.
  - Provided APIs to team 3 and team 4 regarding file system.

- **Future work**
  - Co-work APIs with team 3 and team 4.
  - Address issues with the current implemented APIs.

File system

- Hierarchy



File system: APIs

- GET API Endpoint to Retrieve ETD.
  - services/etds/<int:etd_id>
- GET API Endpoint to Retrieve PDF File.
  - services/etds/<int:etd_id>/pdf
- Save page image (page of .pdf).
  - services/save-page-image
- Retrieve page image with pageID.
  - services/retrieve-page-image/<int:page_id>
- Store the detected objects.
  - services/store-object
- Store the generated XML file.
  - services/store-xml

Database

- **Milestones**
  - Initial Database version comprised of 9 tables.
  - Developed a new Database schema with a set of additional tables.
  - Established relationships between multiple tables for data consistency.
  - Ensured a seamless migration process from the old Database schema to the new schema.
  - Provided support to other teams regarding Database-related queries and issues.
  - Ensured successful data population of ETD and ETDs_metadata tables.

Database Challenges

Choosing Data Types:

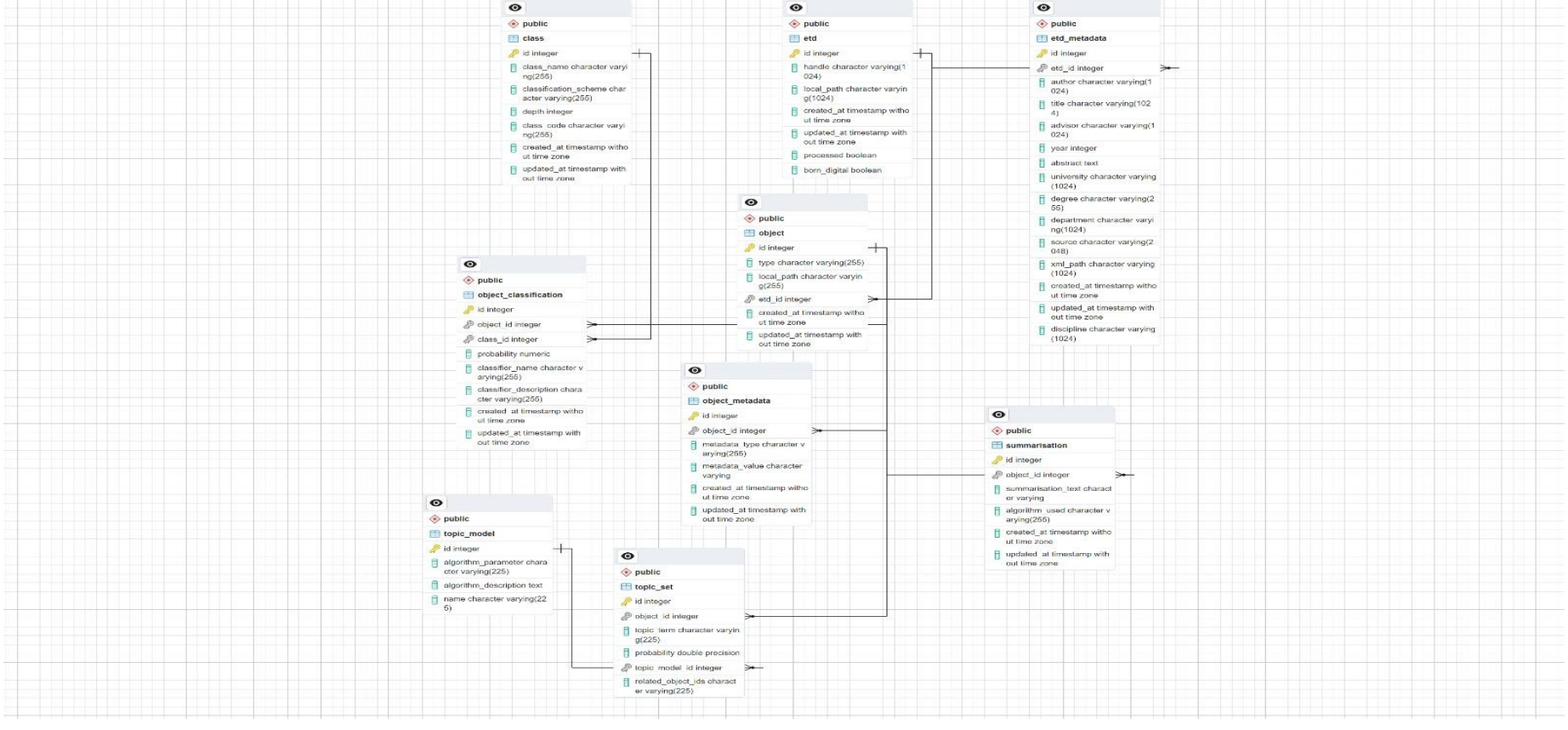
- Encountered difficulties in selecting appropriate data types for columns during tables creation.
- Resolved by collaborating with various teams to understand specific data needs and requirements.

Database Update:

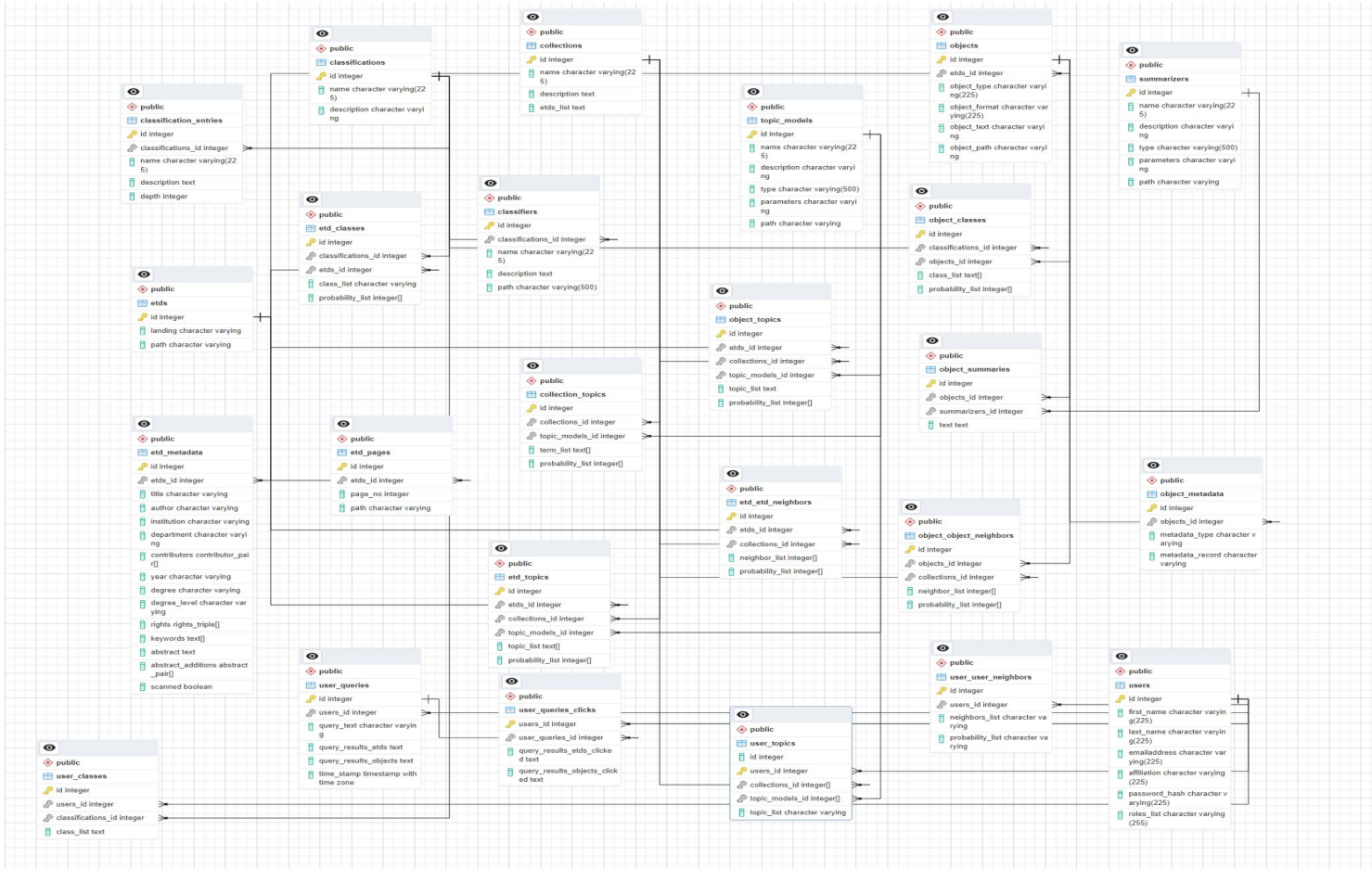
- Updating the Database with 500k records was time-consuming due to the data volume. Developed efficient scripts that significantly reduced the time required for data population.

Old Database Schema

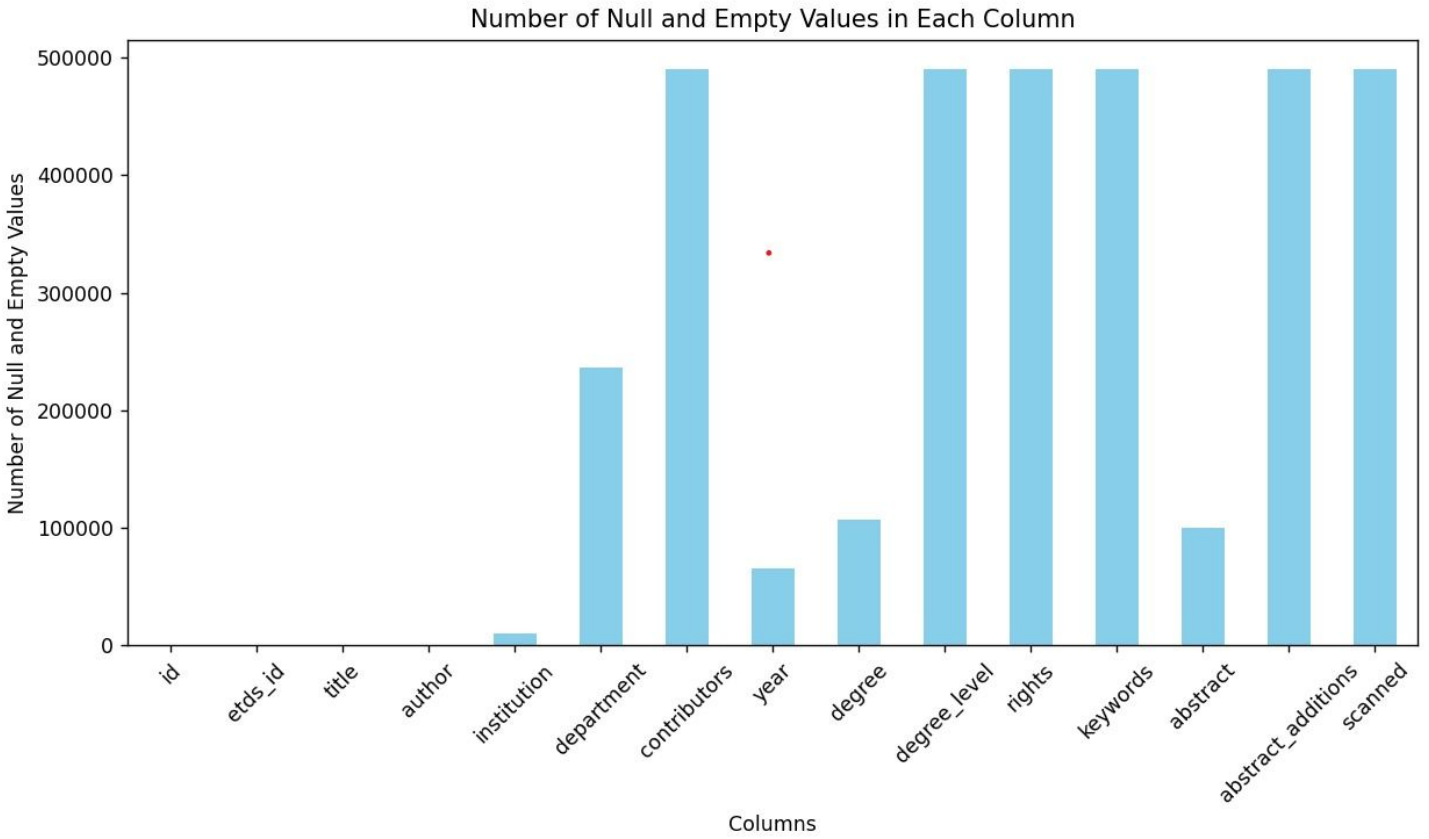
The below image represents the existing version of the Database schema and how the tables are related to one and other.

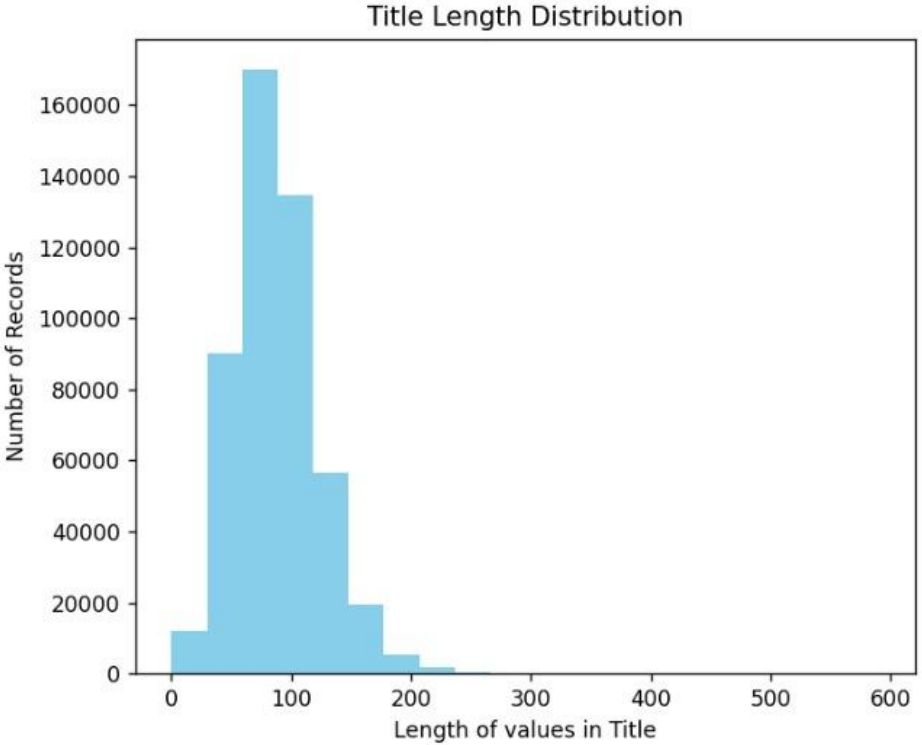


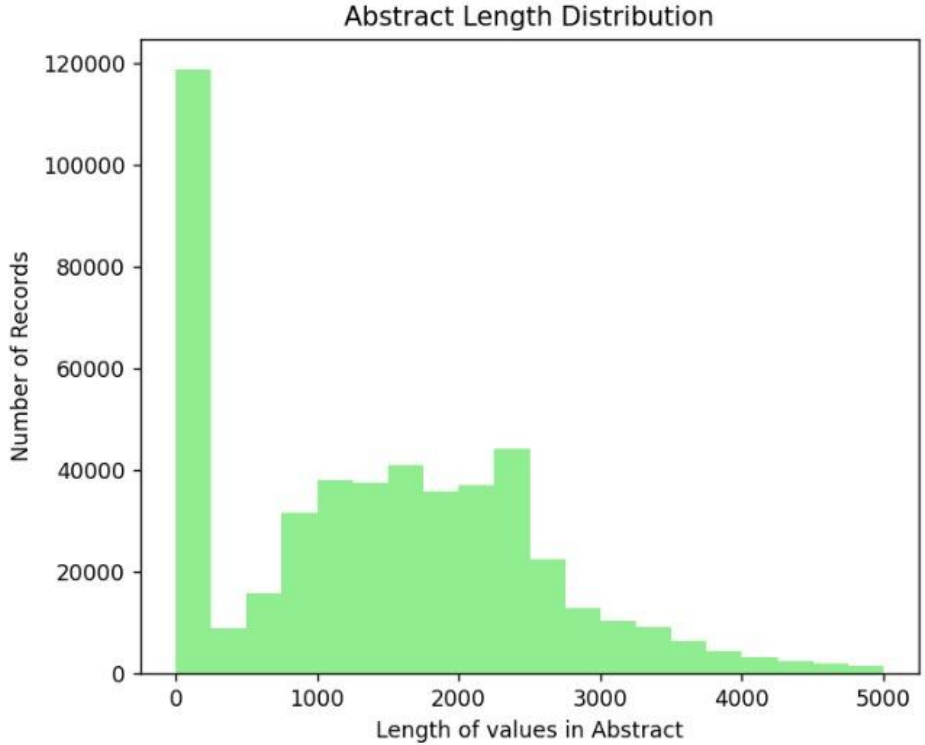
Current Database Schema



Database Statistics




Title Length Distribution
Number of Records
0
20000
40000
60000
80000
100000
120000
140000
160000
0
100
200
300
400
500
600
Length of values in Title


Abstract Length Distribution
Number of Records
0
20000
40000
60000
80000
100000
120000
0
1000
2000
3000
4000
5000
Length of values in Abstract

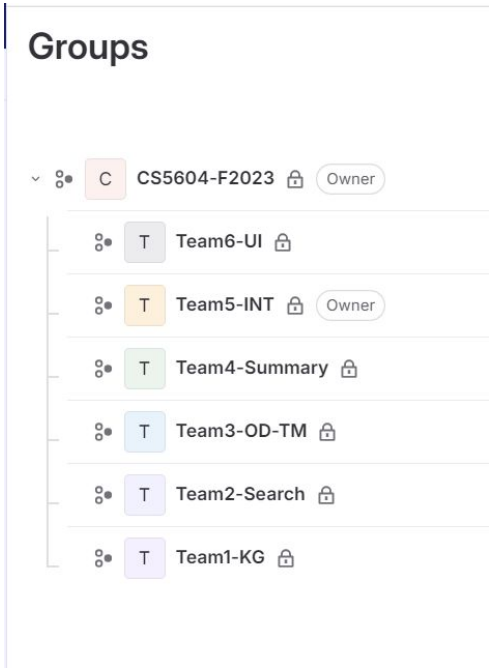
CI-CD

Milestones accomplished:

- Create a group runner for the project.
- Establish connection between group runner and Gitlab code-repository.
- Create group token for running CI-CD setup.
- Create sample CI-CD guide.
- Test deployment pipeline that builds, tests and deploys to endeavour cluster.

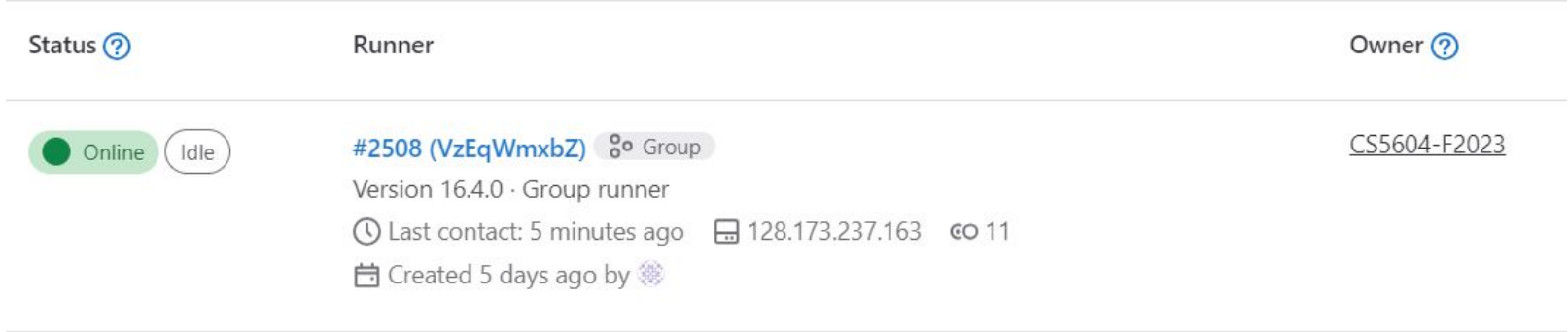
CI-CD

- Gitlab Group to host codebase for all teams

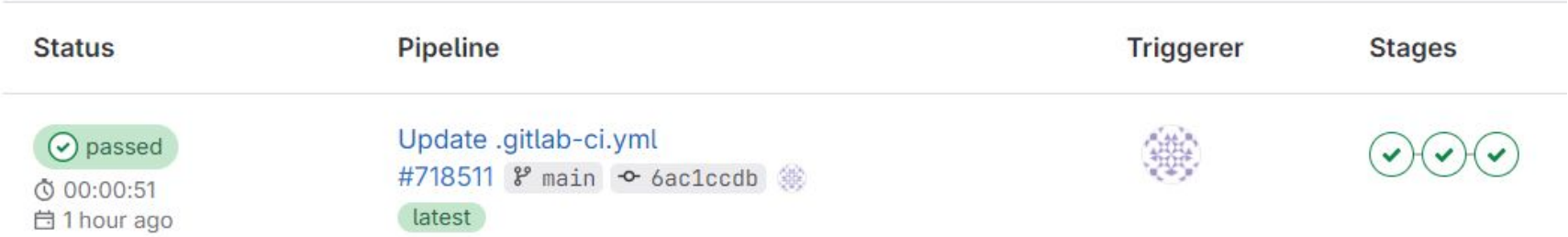


CI-CD

Group runner for the project that is connected to the Gitlab code-repository.



- Test deployment pipeline to deploy to endeavour cluster.



Future Work

- Implement CI-CD pipeline for Database.
- Change the data types of the columns in the Database as per the requirement of other teams.
- Address issues with the current implemented APIs.

Thank You!

Class of F23’
Dr. Fox
Satvik Chekuri
Xiao Liang
Danush Dinesh