5. TAs: Phanindra and Nytik

Problem Statement-5: Back Up service using docker and Kubernetes

General Guidelines for the project:

- Create a GitHub Repository.
- Naming convention: <SRN1>_<SRN2>_<SRN3>_<SRN4>_Project title (srn in ascending order).
- Weekly progress according to the problem statement assigned must be pushed to the repo and this will be considered while evaluating.
- Each team's weekly progress update in the repository must be shown to the teachers in class.

Description: Creating a backup service that periodically backs up the contents of a folder to Google Drive using Docker and Kubernetes involves several steps.

In this project, you will work with Docker and Kubernetes to create a Backup service.

Pre-Requisites/ Pre-Installation:

Docker (Windows | Ubuntu | MacOS)

Kubernetes (Windows | Ubuntu | MacOS)

Deliverables:

- **Week-1: Containerized Google Drive client**

Here's a high-level technical breakdown:

1. **Set up Google Drive API**:
    - Obtain credentials for the Google Drive API.

- Use the `google-api-python-client` library to interact with Google Drive.

2. **Create a Docker Container**:

   - Write a `Dockerfile` that includes all necessary dependencies and your backup script.
   - Build the Docker image.

3. **Write the Backup Script**:

   - Develop a script in Python that uses the Google Drive API to upload files.
   - Ensure the script can be triggered at regular intervals.

❖ **Week-2: Kubernetes Deployment & Orchestration**

- **Kubernetes CronJob**:

  i. Define a `CronJob` resource in Kubernetes to schedule the backup operation.
  ii. The `CronJob` will run the Docker container at specified intervals.

- **Persistent Volume Claims (PVC)**:

  i. Use PVCs in Kubernetes to ensure the data you want to back up is accessible to the container running the backup script.

- **Monitoring and Logging**:

  i. Implement logging to track the backup process.
  ii. Optionally, set up monitoring to alert you in case of failures.

- **Security Considerations**:**Testing and Validation**:

  i. Securely manage API credentials and sensitive data.
  ii. Use Kubernetes secrets to store sensitive information.

- **Testing and Validation**:

  i. Test the backup process thoroughly to ensure data integrity.
  ii. Validate the recovery process from the backups.