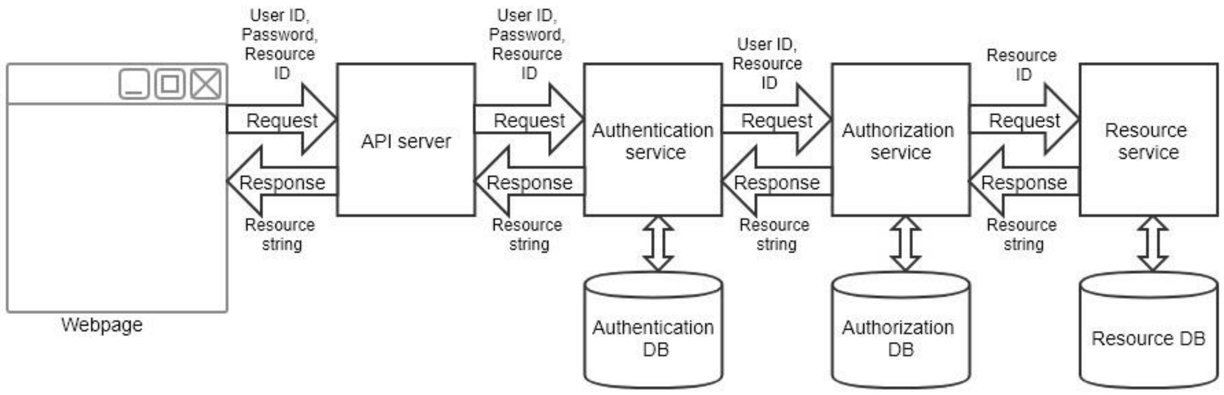
Assignment 1

Problem statement

- Designing and developing an application using the microservice architecture
- Developing 3 microservices in different languages and making them talk with each other

Solution method

- Use-case: An application that authenticates and authorizes a user's access to secret information
- 4 microservices:
    - **API gateway**: This service provides user interface to the user to enter the credentials and shows the file string. It routes the user requests to the appropriate service. It was implemented in NodeJS.
    - **Authentication service:** This service performs user authentication. It accepts user credentials (username and password) and returns true or false. The service was implemented using Python Flask and used a MySQL database to store the authentication information.
    - **Authorization service:** This service performs authorization by checking whether a user is authorized to access a file. It receives a user ID and a file ID as input and returns true if the user is authorized to access a file, and false otherwise. The service was implemented using NodeJS and used a MySQL database to store the authorization information.
    - **Resource/file service:** This service stores all the files and provides read access to them. It receives a file ID and returns the file string if it exists. The service was implemented using Spring Boot and used a MySQL database to store the file strings.



Evaluation and Recommendations

- The services were created by following the microservice principles
    - Independent data-stores
    - Exposed APIs to interact, no interaction using databases
    - Each service could be scaled and maintained independently
- Disadvantages

- App deployment was difficult and produced different problems on different environments
- Scaling and coordination of services was difficult
- No continuous deployment and integration

General technical recommendations

- Microservice architecture provides several benefits like:
    - Small size of code that enhances maintainability
    - Fault isolation

Contributions

- Wiki
- Code

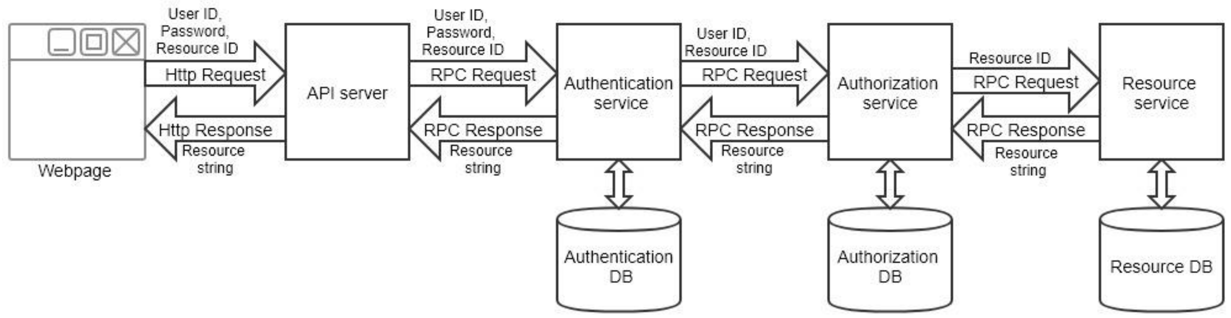
Assignment 2

Problem statement

- Extending the above solution to have easy and reliable deployment
- Implementing CI/CD
- Using a standardized way to interact between the services

Solution method

- Containerization
    - Used Docker to containerize the microservices
    - Make the application deployment easier and reproducible
    - Much faster than full-fledged VMs
    - More secure than deploying multiple apps on the same machine
- Continuous integration / Continuous deployment
    - Used Jenkins to perform CI/CD
    - Github published commit event to Jenkins
    - After getting notified from about a new commit from Github, Jenkins performed build on the latest code and deployed it on the appropriate servers
- Messaging
    - Used RabbitMQ to interact between different microservices
    - Used RPC model of RabbitMQ



General technical recommendations

- Containerization minimizes app setup time and thus provides benefits like faster onboarding of new teammates, reproducible results, etc
- CI/CD helps developers to get a hold of what fails earlier in the development process

Contributions

- Wiki
- Code

Assignment 3

Problem statement

- Developing a Python Jupyter Notebook API to launch experiments

Solution method

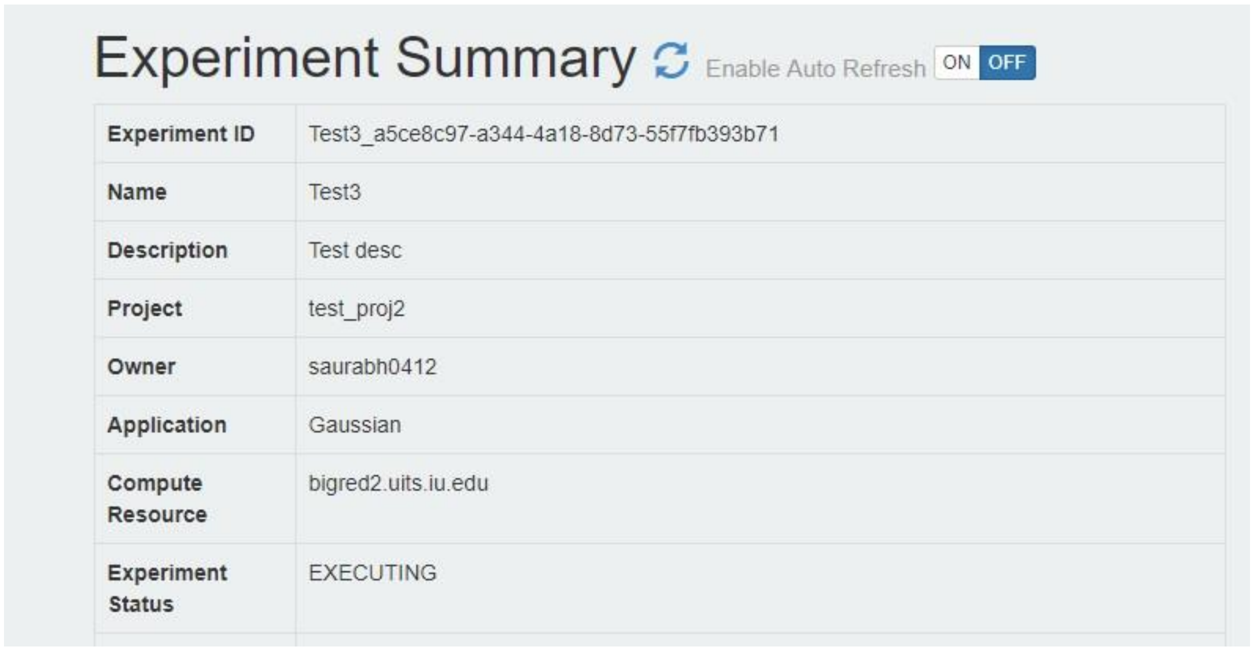
- Used Thrift files from Airavata Python 3 client
- Developed a Python API for launching an experiment
- Developed a test script to test the API

Was this adopted?

No, this API was just a cover around the Airavata Thrift API. The goal of this assignment was to understand Airavata, Airavata API, and its components.

Contributions

- Wiki
- Jira
- Code


Experiment Summary Enable Auto Refresh ON OFF

Experiment ID	Test3_a5ce8c97-a344-4a18-8d73-55f7fb393b71
Name	Test3
Description	Test desc
Project	test_proj2
Owner	saurabh0412
Application	Gaussian
Compute Resource	bigred2.uits.iu.edu
Experiment Status	EXECUTING

Assignment 4

Problem statement

- Finding a technology that:
    - Is a standard way to represent workflows
    - Has free tools like GUI (Rabix) for creating workflows, an execution engine, etc
    - Will be able to integrate with Airavata

What I did?

- Evaluated the technologies: CWL and Airflow for representing Airavata workflows on the above requirements
- Read about Airflow and discovered that it is not a suitable technology for implementing workflows [Jira comment]
- Read about CWL and discovered that CWL fulfills all the requirements

Was this adopted?

- Yes, we continued with CWL for implementing Airavata workflows

Alternative solutions

- Pinball by Pinterest

General technical recommendations

- Best technology is the one that provides the best fit even if it is not the coolest technology in the market

Contributions

- Wiki
- Jira
- Email

Assignment 5

Problem statement

- Evaluating Common Workflow Language
- Understanding how Airavata workflows used to work
- Understanding CWL syntax

Evaluation and recommendations

- Observations about CWL:
    - Common Workflow Language is an open specification for representing workflows
    - It is slowly picking up as a standard among the scientific community for sharing workflows
- How Airavata workflows used to work [Jira comment]?
    - Experiments in Airavata are a set of tasks
    - Thus, we represent workflows using a DAG of tasks
    - We used XWF (Xbaya Workflow Language) files to represent an Airavata workflow
    - These files were created by the Xbaya GUI integrated in Airavata
    - Registry stores details about the workflow tasks
- Observations about CWL syntax
    - CWL provides a rich vocabulary for representing different aspects of a workflow like different datatypes for input & output, file handling, environment variables, expressions, etc

Solution method

- Installed CWL and CWL tool
- Executed several workflows to evaluate the feasibility of CWL to represent Airavata workflows

Contributions

- Wiki
- Jira

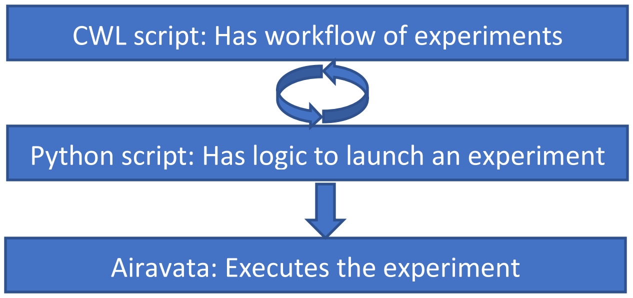
Assignment 6

Problem statement

- Executing a workflow of experiments using CWL

Solution method

- Analyzed feasibility of using CWL to represent Airavata workflows by executing test workflows
- Developed a Python script to launch an Airavata experiment
- Developed a CWL script that executes a workflow of experiments by repeatedly calling the above Python script


CWL script: Has workflow of experiments
Python script: Has logic to launch an experiment
Airavata: Executes the experiment

Evaluations and recommendations

- CWL can represent Airavata workflows

Contributions

- Wiki
- Jira
- Code

Problem

- Since experiments are non-reusable units, we need to create the experiment manually before executing them using CWL

Assignment 7

Problem statement

- Creating experiments using CWL

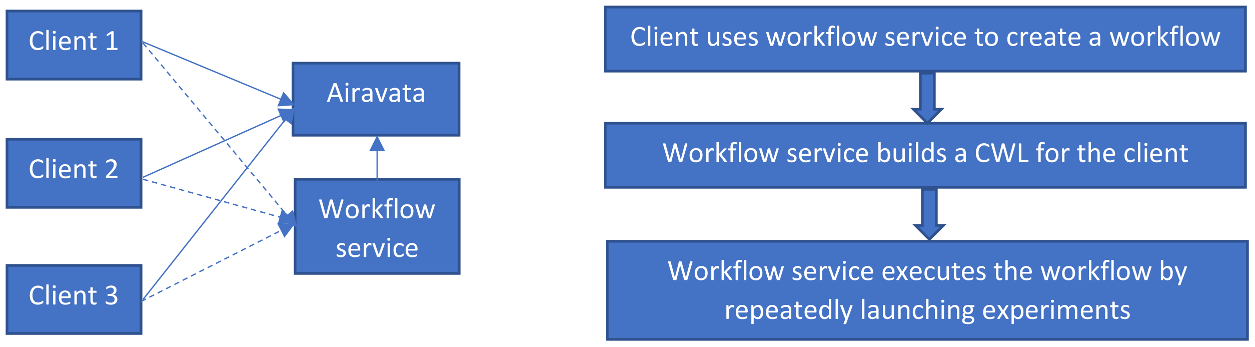
Solution method

- Enhanced the Python script to perform experiment creation before launching it

- Modified the CWL script to perform creation according to the new Python script

Evaluations and recommendations

Current architecture


Client 1
Client 2
Client 3
Airavata
Workflow service
Client uses workflow service to create a workflow
Workflow service builds a CWL for the client
Workflow service executes the workflow by repeatedly launching experiments

Current architecture

Contributions

- Wiki
- Jira
- Code

Problems

- We are still unable to take CWL script as input and execute a workflow
- If we want to input a CWL script, we need to write a custom parser that translates CWL into Airavata experiment object
- Writing a CWL parser will not be an easy task