Stack Overflow backend: From monolith to microservices

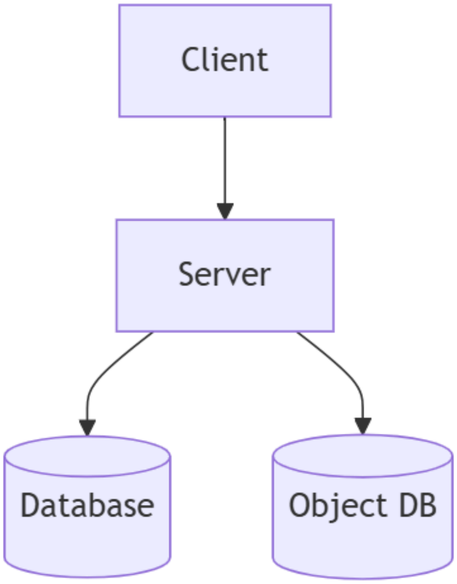
Goal

Your task is to build a monolithic web based system. Then, break down the system into microservices following the microservice architecture. To achieve this follow the steps below.

Part 1: Build a server and a client

In this part, you have to create a basic version of Stack Overflow. The app itself should not be complex as we are here to build a distributed system and not a full featured online community for developers.

Architecture



Requirements

Client Requirements

The client UI will have the following routes:

- SignUp page
- SignIn page
- Home page: should show a list of posts (texts/code snippets)
- Notification page: should show notifications of recent posts
  - Clicking on the notification should show the post

Server requirements

APIS

- /signup endpoint for registering new users
  - user signs up with email and password
- /signin endpoint for signing into the system
  - user signs in with email and password
- /post endpoint for creating and retrieving posts
  - GET: Get latest posts of all users except logged in user
  - POST: Create new posts for user
- /notification endpoint for creating and retrieving notifications
  - GET: Get notifications
  - POST: Create notification against a post

Jobs

- Notification cleaner: A job should periodically check for old notifications and delete them

System assumptions

- The system needs no other services (e.g. comments, votes etc.)

Other instructions

- Code snippet uploaded with posts should be stored in an object store database (MinIO)
- You may use any programming language. However, Node.js or Python or Go is recommended.
- Frameworks/libraries can be used. However, using frameworks (like express/flask/fastapi/gin) will make the assignment easier.
- Using a framework on the frontend can be useful (angular/vue/react).
- You can use any database.
- Don’t implement extra features. Because, distributing them in multiple instances could turn out to be a problem later.