Hasty

Backend Case Study

1 - 2 days

Background

Hasty is a next-generation vision AI platform that brings the model into the process of creating custom, proprietary applications. This empowers our users to get their advanced applications to market significantly faster and makes their solutions more robust.

The backend is quite complex, consisting of more than 20 services, orchestrated by Kubernetes. Some of the services require real time response, others are done in a batch-processing manner. The goal of this challenge is to allow candidates to demonstrate their skills and understanding of engineering asynchronous jobs processing in the most efficient and scalable way.

Task

Problem

Imagine you have a process that takes some time to complete (say, some complex report). Users can trigger its execution using REST API, and will be somehow notified after it's done.

The process should look like this:

- User performs a PUT/POST request with some object_id, receives a job_id.
- This job will then be executed asynchronously (sleep command can be used with a random time from 15-40 seconds to simulate this).
- It's up to you as to whether this async job processing will happen in the same API server or in a separate service, but you should be prepared to explain why you went down that path and why you think it's preferable to the other option.
- The execution results should be persisted somewhere (timestamp, object_id, job_id, sleep time used).
- Users can then perform GET requests with job_id to get its associated metadata, especially the status (which can be a string).

Requirements to the service:

- Tasks with the same object_id should be executed only once in a time window of 5 minutes.
- It should rerun/resume any incomplete jobs if service goes down or have a restart.
- The job should be canceled if it exceeds a preconfigured timeout.
- Must be horizontally scalable.

Hasty

Implementation requirements:

- Can be implemented in Go/Python.
- Should be run in docker containers.
- Have e2e tests coverage.

Evaluation

Your challenge results will be evaluated based on the following criteria:

- Code cleanliness and correctness.
- Level of creativity / ingenuity of your solution.

Tips

Conciseness is appreciated. The goal is to see your approach and thinking - not to make you work late nights building detailed solutions

- Feel free to make assumptions where necessary
- The submission will be used as a basis for the discussion in the next interview

Enjoy and please let us know if you have any further questions! We're looking forward to your results.