GL5 Project

Requirements

Write a distributed application that you are passionate about. The application should fulfill a well defined task that's useful (not something that calculates the factorial of the number).

Having a UI is not strictly required, a working API is enough.

We require the application to rate limit user requests based on their IP address.

DevOps

- Observability:
    - Metrics:
        - The application/service should have some general purpose metrics (mem, cpu, request count) exported.
            - 1 metrics that allow us to track how many request we are processing
            - Label the metric in a way that allows you to get the success ratio (% of successful requests)
            - Label the metrics in a way that allows you to know many requests each http route processed.
        - Business logic metrics
            - Add a metrics that reflect a business value of your application, example:
                - Order count per item (for a shop website)
                - Friend request rejected (for a social media application
    - Logs:
        - All logs emitted when processing a request should have the attribute request_id that uniquely identifies the request
            - For any given request you should either:
                - Generate a new request id (an UUID), or
                - Use whatever value found in the request header X-Request-ID
        - All logs should have the attribute client_ip which is the ip address of the caller
    - Trace:
        - Generate traces for your handlers
        - Add the same attributes you been asked to add in logs to the traces as well (request_id and client_ip)
- helm:
    - Write a helm chart for your service
    - Make these values configurable via the file values.yaml
        - image.name
        - image.tag

        - http.port
        - deployment.replicas
    - Make it possible to add arbitrary `labels` to the deployment via the file values.yaml
    - Make sure you define the correct env variables so that open telemetry tracing work as expected

Automation

- Automated infrastructure provisioning
- Automated initialization for workloads
- Multi-environment setup (Or at least the project should be capable of supporting extra environments)
- Well isolated and maintainable infrastructure layers (Microstacks)
- Automated Deployment

Deployment

- Usage of Microservices
    - Use all good practices of tracking workloads health and restoring them when needed
- Adequate choice of deployment strategy (+ explanation)
    - Note: Recreate and Rolling updates are not accepted
- Use of required configurations and secrets (treat those secrets carefully)
- Use an adequate network architecture and a well-defined set of network policies
- Have at least one endpoint exposed to the public

Shared

- Code quality
    - Well structured project tree
    - Good documentation, project stack overview + guides
- All work should be available in a git repository that we (Wassim and Meher) can access.
    - Project setup should be straightforward and predictable(we will build it on our machines), Use the adequate technologies to ensure that.
    - The repository should contain the
        - Application source code
        - The helm chart definition
        - Files needed to build the application
        - Documentation where the application architecture is well explained.
- We prefer each student to work individually, maximum is 3.
    - **The bigger the group the higher the expectations.**