Technical Assessment Test

This document contains recruitment tasks in a form of mini-quiz designed to be used in Fluency recruitment process for technical positions. If you are the person who received it and you did post your candidacy to Fluency - congratulations on getting into the next stage! Some of them might require writing down a few paragraphs [in English], some are purely technical mini-coding exercises that might be done in any programming language. Please take your time and do your best answering as many of them as you can! All the questions here should take you no more than a few hours to answer.

If you did not take part in Fluency recruitment process and have no idea why you got this document, then please ignore it. Under no circumstances, do NOT publish or send anyone the contents of this document by yourself.

Good Luck!

1. Fixing the code:

Convert the code below using any lib of your choice to be more readable:

```
service.fetchData(function (err,data) {
  if(err){
    console.log(err)
    return
  }
  data.owner = "Fluency"
  service.parseData(data, function (err,data) {
     if(err){
        console.log(err)
        return
     }
     service.saveData(function (err,data) {
       if(err){
          console.log(err)
          return
       }
      console.log("finished")
     })
  })
});
```

2. Fixing numerical errors:

Program below outputs a value of 0.020000000000000004 instead of expected 0.02. Can you determine what is the reason for this "strange" behaviour? How does arithmetic calculations are performed in JavaScript? What kind of tools would you suggest using for performing calculations which require absolute precision?

```
var x = 0.1 * 0.2;
console.log(x);
```

3. Fixing deployment:

Imagine you received a deployment configuration for docker-compose as the one presented below and you are asked to convert it for Kubernetes YAML configuration for developers to use locally in their minikubes for development purposes. How would newly created K8S configuration look like?

```
version: "3.7"

services:
 api:
   build:
     context: .
   image: "fluency-engine"
   container_name: "fluency_engine_api"
   env_file: docker-compose.env
   environment:
     SERVICES: api
     PORT: 3000
   depends_on:
     - nats
   labels:
     - "traefik.enable=true"
   networks:
     - internal
   restart: on-failure

 nats:
   image: nats:2
   networks:
     - internal

 traefik:
   image: traefik:v2.1
   command:
     - "--api.insecure=true"
   ports:
     - 3000:80
     - 3001:8080
   volumes:
     - /var/run/docker.sock:/var/run/docker.sock:ro
   networks:
     - internal
     - default

networks:
 internal:

volumes:
 data:
```

4. Fixing Infrastructure

Imagine you are working on a microservice-based application that provides some kind of usability, implemented in a form of multiple different services and replicas working together by messaging pattern. At first everything is going smoothly, but as the system continues to grow more and more, communication between services become more complex due to convoluted dependencies. The error-handling and debugging starts to be especially problematic as there is no central coordinator to take this responsibility on itself. What’s even worse, partial failures halt the system as it's hard to make proper rollbacks. What kind of orchestration and error-handling patterns would you suggest for developers in that case, to make their app easier and solve those problems?