项目要求

负载均衡

1. 多台机器共同支撑用户访问，机器发生单点故障，在可用性要求范围内可将请求转发到正常机器。
2. 机器故障恢复后再度恢复访问支撑能力

不间断服务

7*24 小时不间断服务

数据吞吐及并发

1. 基础配置下2000/s 并发请求支持
2. 服务平均响应时间50ms以下
3. 吞吐量可横向扩展，且平均响应时间不发生明显波动
4. 10亿+数据处理能力

可用性

99.99% 高可用

环境要求

服务端

硬件

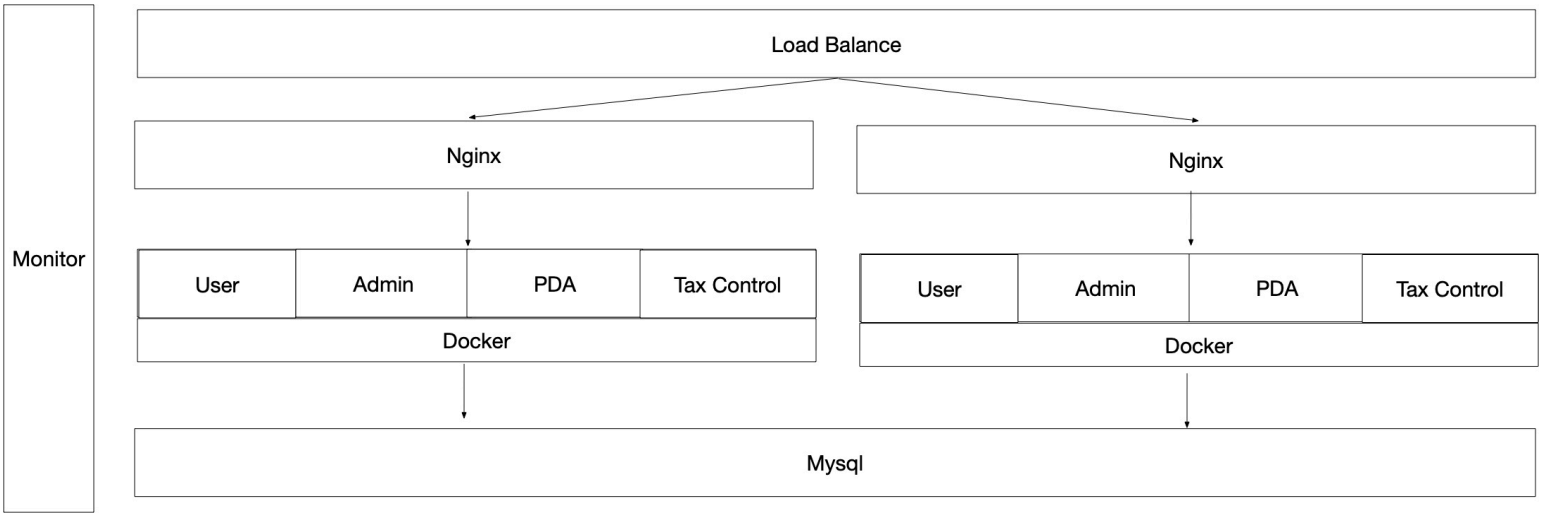
1. 2 * 2CPU 8g 内存 硬盘存储 不低于 100GB（负载均衡）
2. 2 * 数据库 4c 8g 1T （高可用、备份）

软件

1. 操作系统 Linux Centos 8 及以上
2. jdk 1.8 及以上
3. mysql 8.0 及以上

4. docker 20 及以上

部署架构



部署方案

可采用docker-compose/k8s 将方案脚本化，启动指令一键完成环境构建

防伪税票唯一编码说明

1. 原始编码采用首字母+序列号的编码方式,首字母代表具体的产品 A/W/B 分别代表 Alcho,Wine,Beer
2. 对外展示码，根据防伪税票的最后一位的值获取防伪税票的加密位，然后根据加密规则配置生成对外展示码
3. 可根据提供的对外展示码，反向验证税票是否是经过合法加密的
4. 附详细参考代码:

```
package com.little.g.springcloud.smileback;

import org.apache.commons.lang3.RandomUtils;

import java.math.BigDecimal;
import java.util.HashSet;
import java.util.Set;

public class TicketUtil {
```

```
    private static char[] mixArray = { 'x', 'N', 'a', 'z', 'R', 'v', 'o', 'X', 'U'
, 'e' };

    private static int[][] mixInt = { { 1, 3, 7, 5 }, { 3, 7 }, { 4, 7 },
            { 7, 4, 3, 5, 1, 2 }, { 5, 3, 1 }, { 7, 4, 1 }, { 5 }, { 6, 7 }, { 2,
4, 7 },
            { 1 } };

    public static String generateTicket(String f) throws Exception {

        StringBuilder sb = new StringBuilder();
        sb.append(RandomNum());

        int pos = Integer.parseInt(String.valueOf(sb.charAt(0)));

        int[] replaceChars = mixInt[pos];

        for (int i = 0; i < replaceChars.length; i++) {
            int replacePos = replaceChars[i];
            int current = Integer.parseInt(String.valueOf(sb.charAt(replacePos)));
            sb.replace(replacePos, replacePos + 1, String.valueOf(mixArray[current
]));
        }

        return String.format("%s%s", f, sb.toString());

    }

    public static String generateRandom() {
        StringBuilder sb = new StringBuilder();
        for (int x = 0; x < 10; x++) {
            int i = RandomUtils.nextInt(1, 8);
            sb.append("{");
            Set<Integer> sets = new HashSet<>();
            for (int j = 0; j < i; j++) {
                int r = RandomUtils.nextInt(1, 8);
                if (sets.contains(r)) {
                    continue;
                }
                sb.append(r);
                sets.add(r);
                if (j != (i - 1)) {
                    sb.append(",");
```

```
                }
            }
            sb.append("}");
            sb.append(",");
        }
        return sb.toString();

    }

    public static Long RandomNum() {
        return new BigDecimal(Math.random() * 100000000 + 100000000).longValue();
    }

    public static void main(String[] args) throws Exception {
        for (int i = 0; i < 1000; i++) {
            System.out.println(generateTicket("F"));
        }
    }

}
```