AgileConfig

CONTENT

01 引入

大厂出品

spring-cloud-config

Java

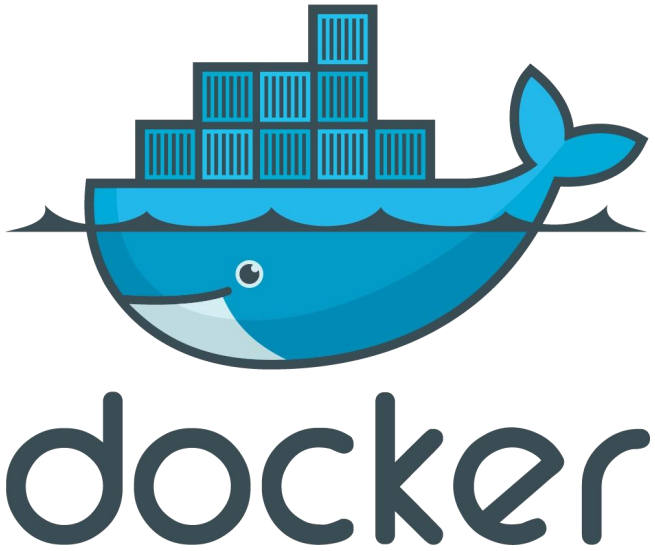
disconf

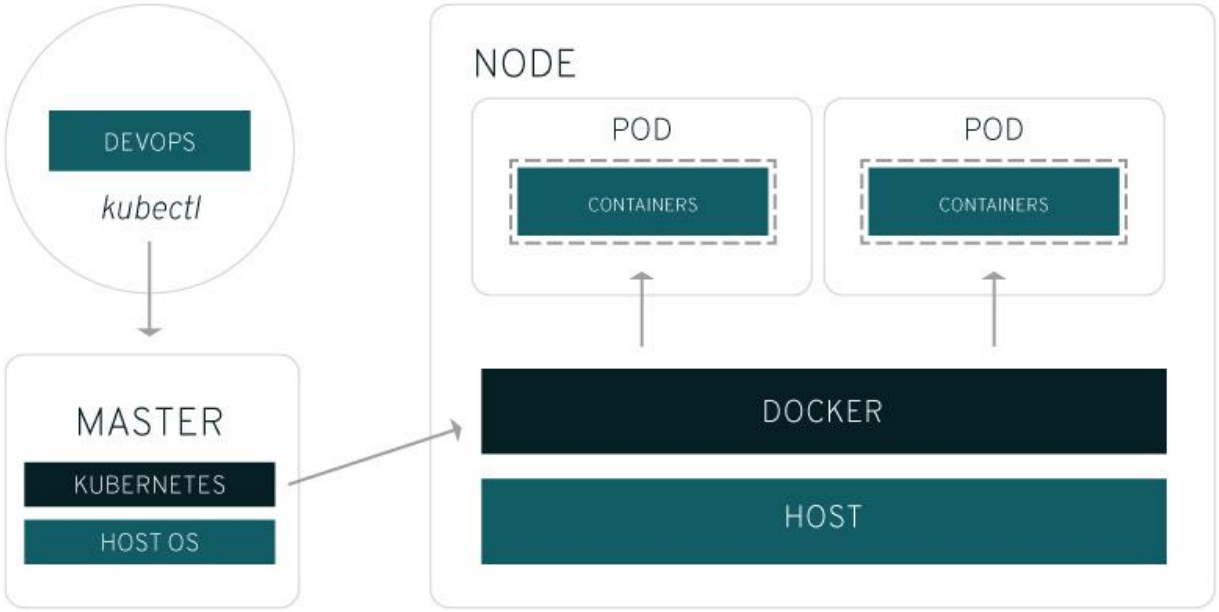
Java

apollo

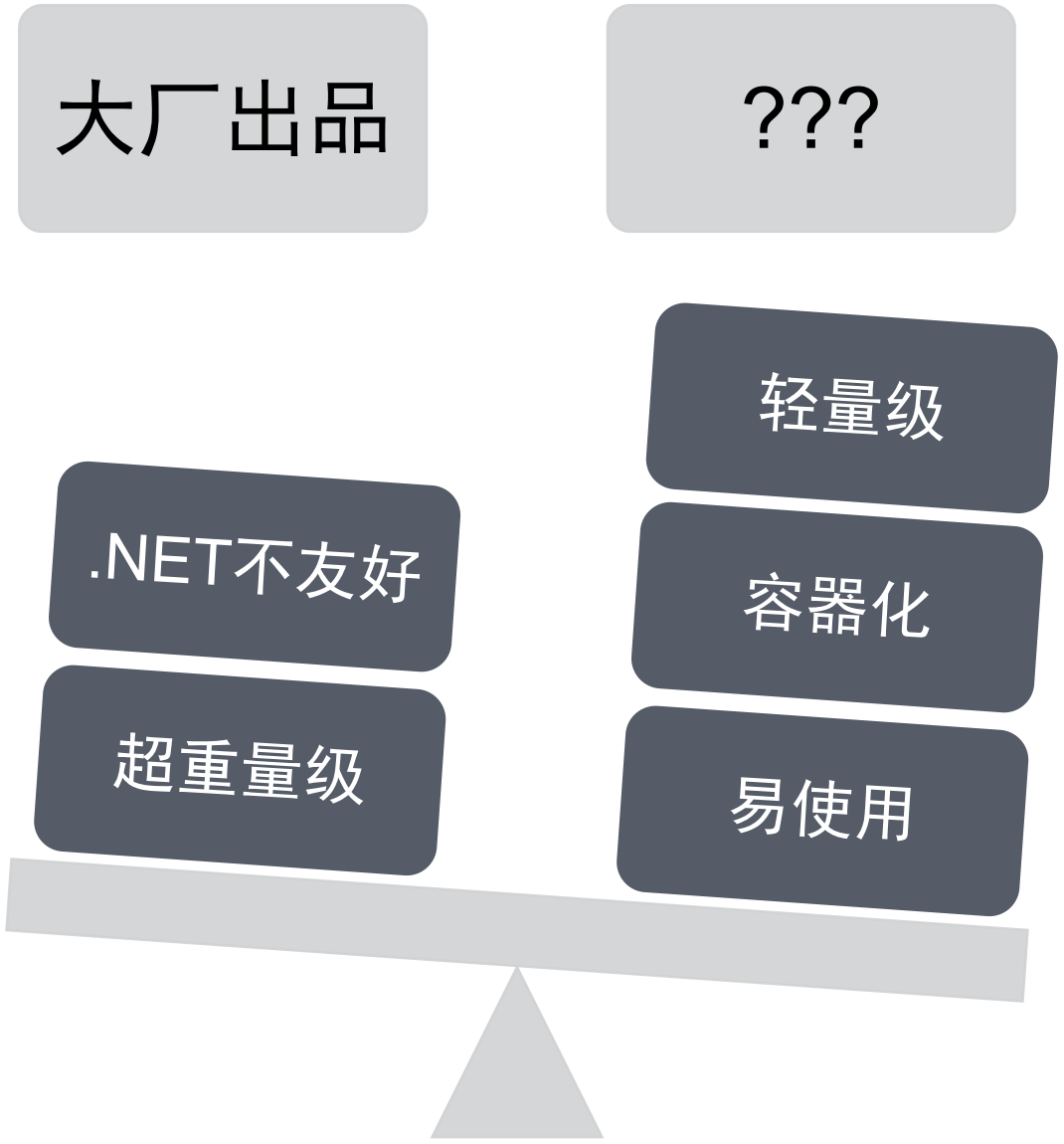
.NET

.....


docker


DEVOPS
kubectl
MASTER
KUBERNETES
HOST OS
NODE
POD
CONTAINERS
POD
CONTAINERS
DOCKER
HOST

产品分析



项目引入

轻量化

高可用

易部署

多存储

平台语言

ASP.NET Core

.NET Core

React

特点

- 部署简单，最少只需要一个数据节点，支持docker部署
- 支持多节点分布式部署来保证高可用
- 配置支持按应用隔离，应用内配置支持分组隔离
- 应用支持继承，可以把公共配置提取到一个应用然后其它应用继承它
- 使用长连接技术，配置信息实时推送至客户端
- 支持IConfiguration，IOptions模式读取配置，原程序几乎可以不用改造
- 配置修改支持版本记录，随时回滚配置
- 如果所有节点都故障，客户端支持从本地缓存读取配置

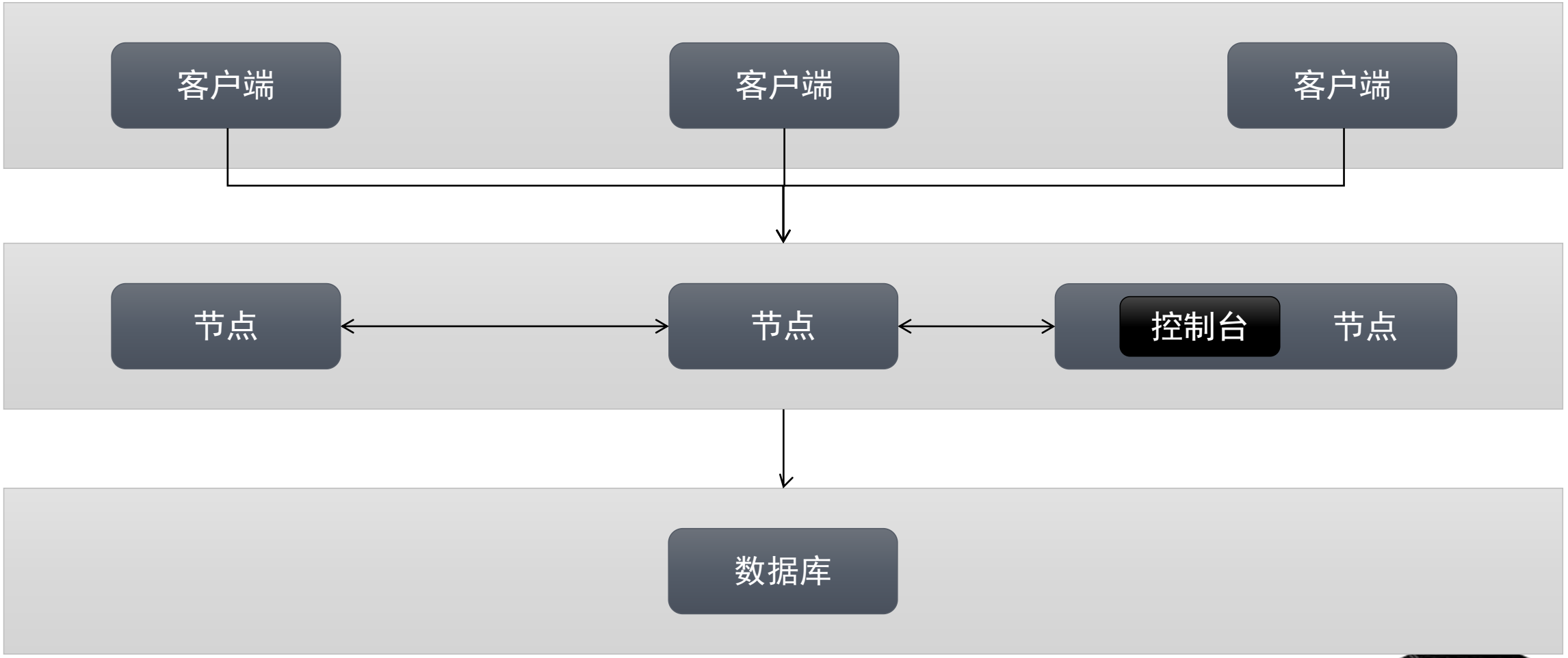
项目地址

Github地址：https://github.com/kklldog/AgileConfig

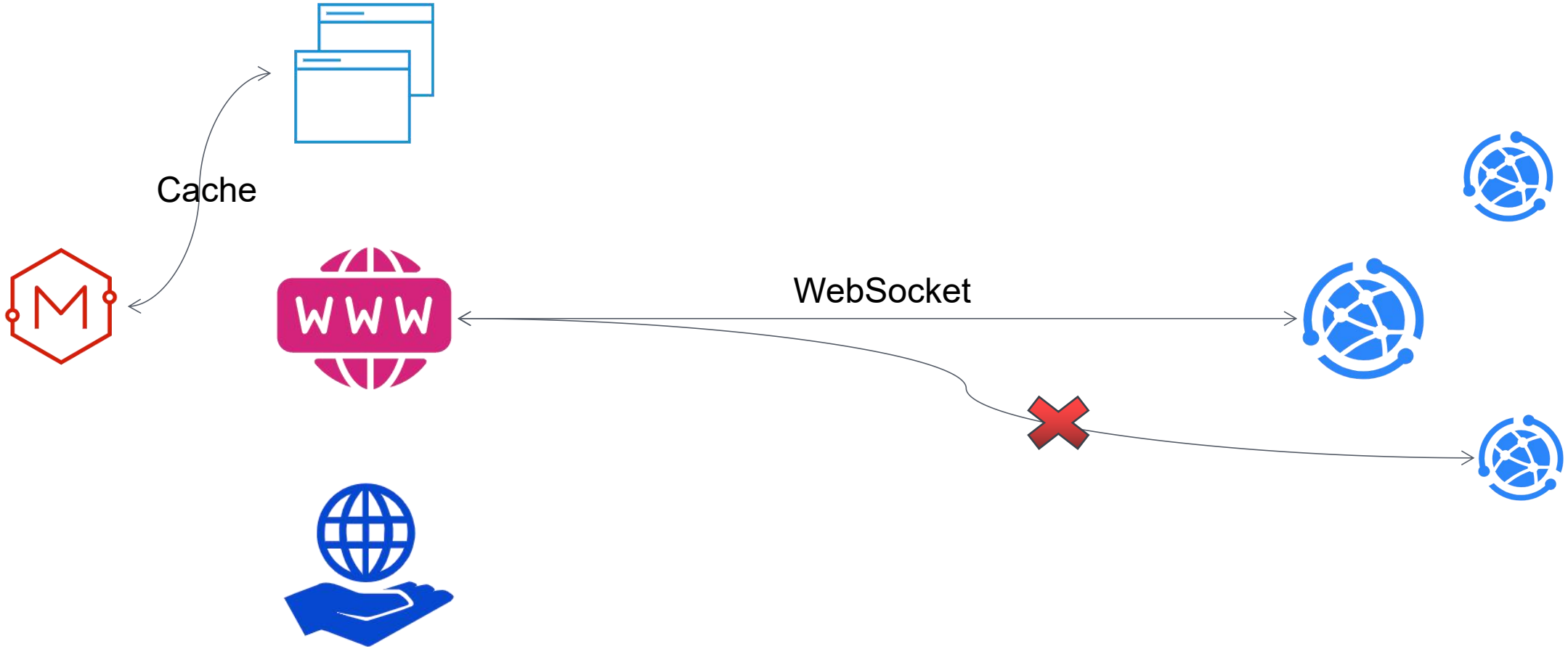
Gitee 镜像：https://gitee.com/kklldog/AgileConfig

02 设计

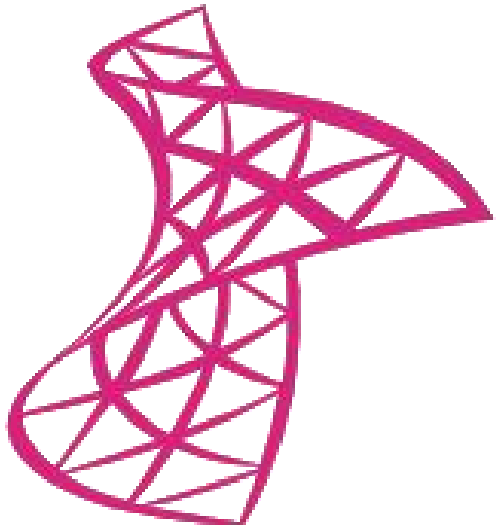
架构



客户端、节点、管理程序



数据库



ORACLE

高可用

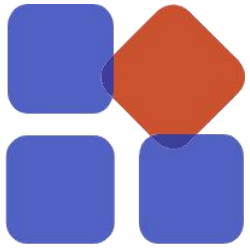
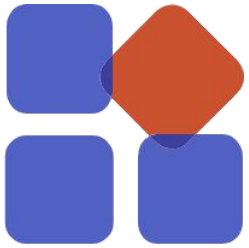
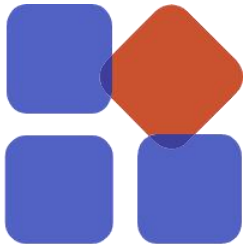
AgileConfig 的节点都是无状态的，所以可以横向部署多个节点来防止单点故障。在客户端配置多个节点地址后，客户端会随机连接至某个节点。

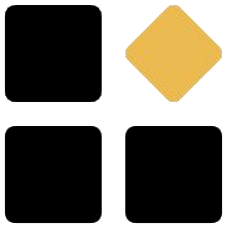
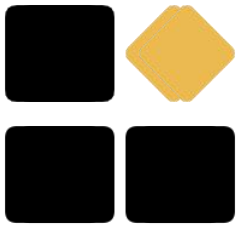
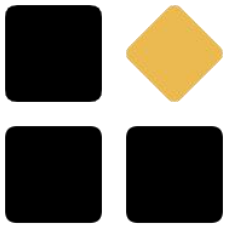
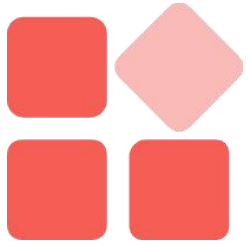
问题	影响	说明
控制台下线	无法维护配置，客户端无影响	因为控制台跟节点是共存的，所以某个控制台下线一般来说同样意味着一个节点的下线
某个节点下线	客户端重连至其他节点	无任何影响
所有节点下线	客户端从内存读取配置	启动的客户端会从内存读取配置，未启动的客户端会再尝试连接到节点多次失败后，尝试从本地文件缓存读取配置，保证应用可以启动

03 剖析

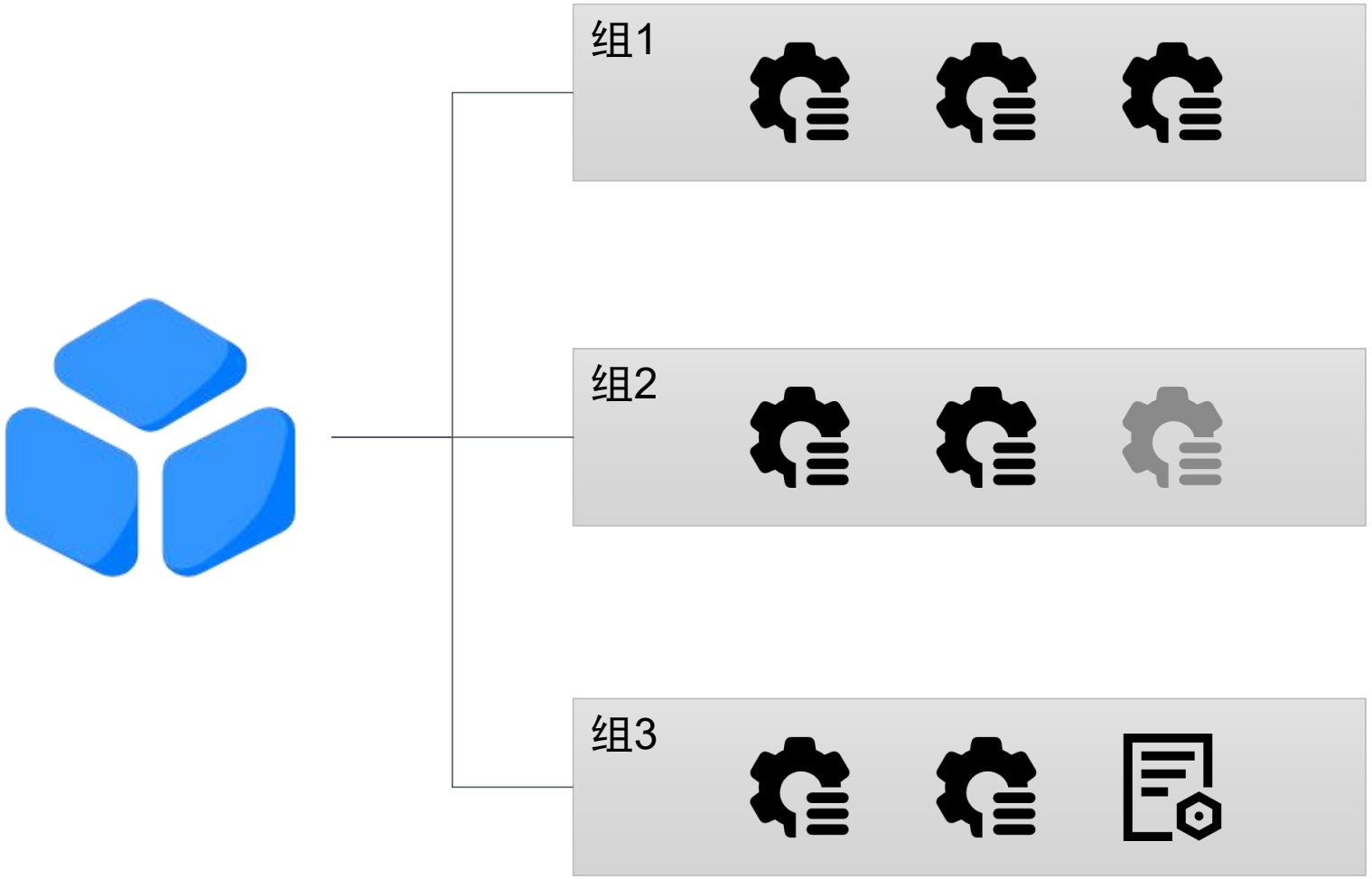
节点

应用


配置项

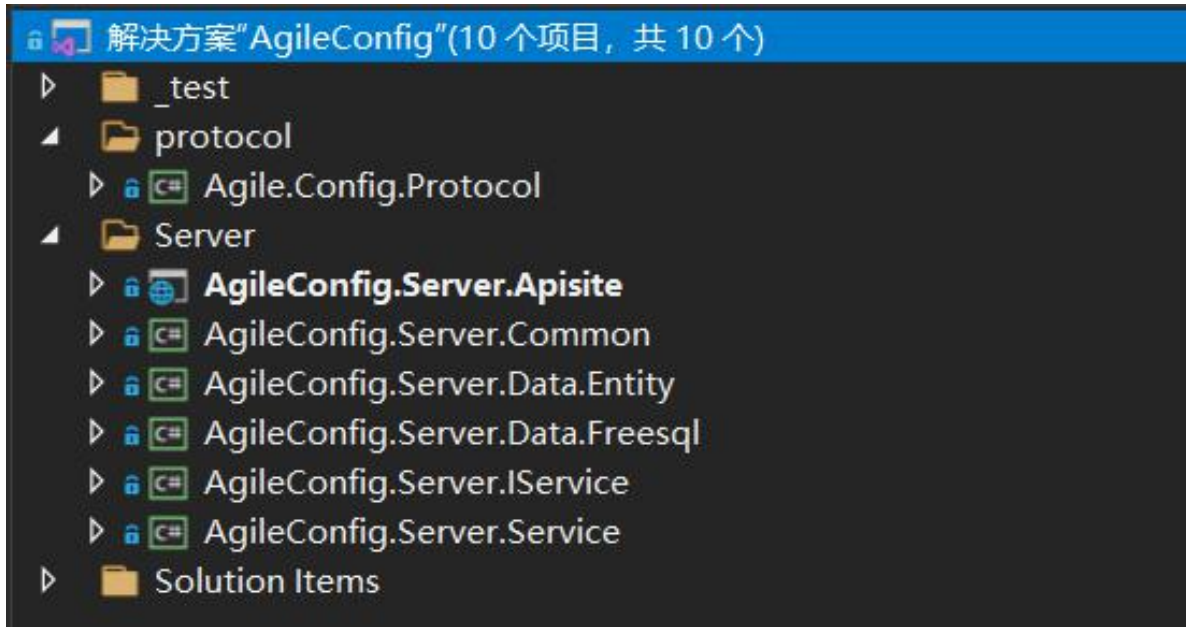


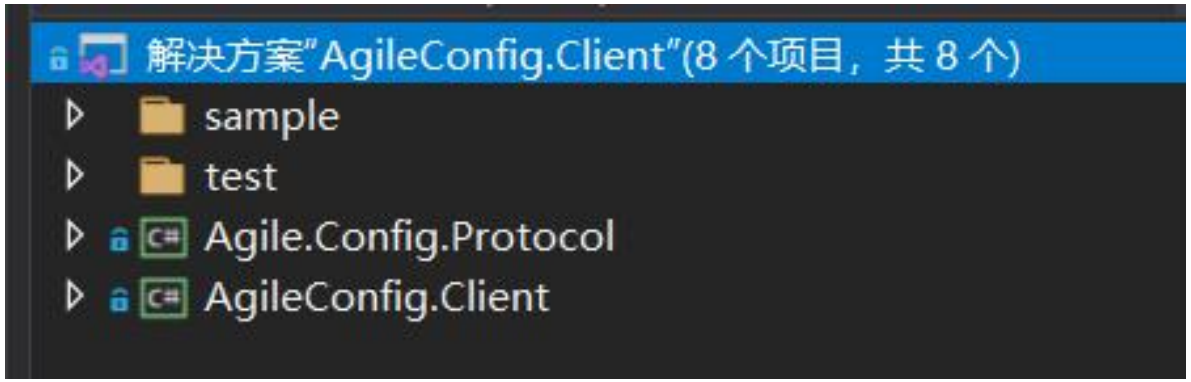
客户端

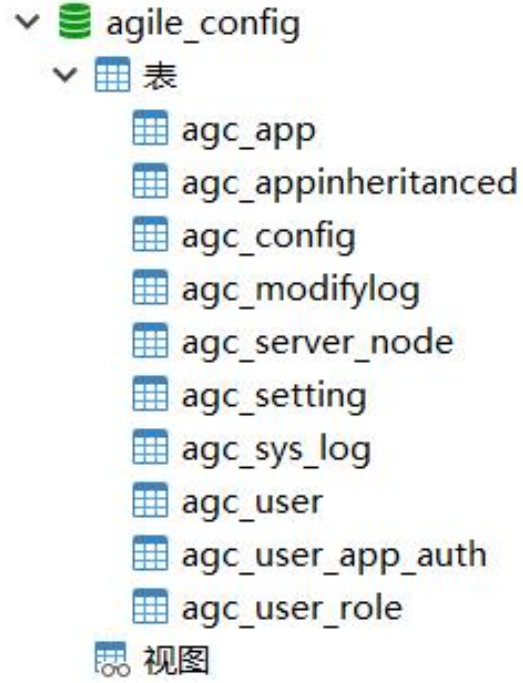
系统日志


AntdvPlus
系统管理员
看板
AGILECONFIG
管理
应用:
类型:
开始日期 ~ 结束日期
查询
应用
类型
时间
内容
0
2021-07-27 09:07:14
admin 登录成功
public_002
0
2021-07-23 10:07:97
用户: admin 回滚配置【Key:11111】 【Group: 】 【AppId: public_002】至历史记录: ead854a2f9a64a3e9ba2afea4d7e29bd
共有 20 条数据
1
2
3
4
5
10

组成


解决方案"AgileConfig"(10 个项目，共 10 个)
_test
protocol
Agile.Config.Protocol
Server
AgileConfig.Server.Apisite
AgileConfig.Server.Common
AgileConfig.Server.Data.Entity
AgileConfig.Server.Data.Freesql
AgileConfig.Server.IService
AgileConfig.Server.Service
Solution Items


解决方案"AgileConfig.Client"(8 个项目，共 8 个)
sample
test
Agile.Config.Protocol
AgileConfig.Client


agile_config
表
agc_app
agc_appinheritanced
agc_config
agc_modifylog
agc_server_node
agc_setting
agc_sys_log
agc_user
agc_user_app_auth
agc_user_role
视图

其他

- EventBus 消息通知机制系统
- 定时节点检测服务
- 数据库及默认数据自动初始化
- 数据版本控制及日志追踪
- Freesql 多数据库支持

04 部署

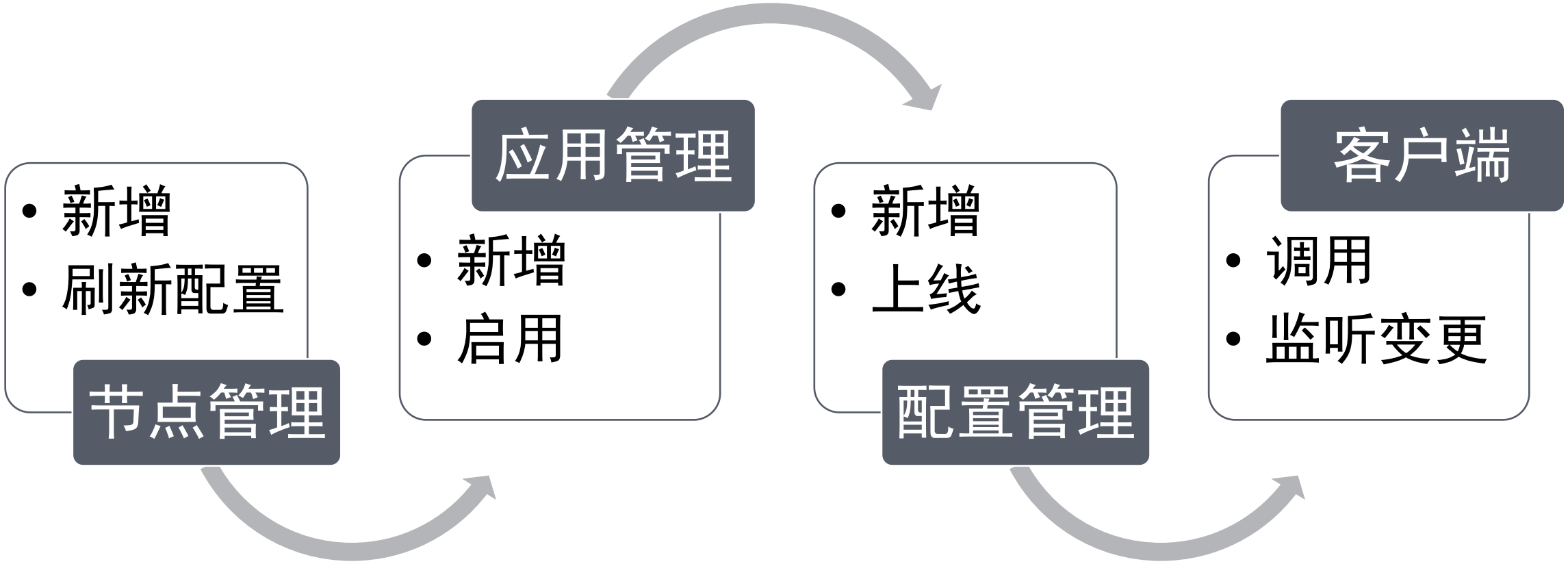
部署服务端

```
sudo docker run --name agile_config -e adminConsole=true -e db:provider=sqlite -e db:conn="Data Source=agile_config.db" -p 5000:5000 -v /etc/localtime:/etc/localtime  kklldog/agile_config:latest
```

参数介绍：

- adminConsole 配置程序是否为管理控制台，如果为 true 则启用控制台功能，访问该实例会出现管理界面，多节点部署的话，只需要开启一个控制台即可。
- db:provider 配置程序的数据库类型，可选 sqlserver，mysql，sqlite，npgsql，oracle。
- db:conn 配置数据库连接串

初始化



05 实例

asp net core web项目示例

安装客户端组件，执行一下命令或者通过Nuget安装。

```
Install-Package AgileConfig.Client
```

然后修改 appsettings.json 文件

```
{
  "AgileConfig": {
    "appId": "LogService",
    "secret": "123456",
    "nodes": "http://localhost:5000,http://localhost:5001"//多个节点使用逗号分隔
  }
}
```

修改 Program.cs

```
public static IHostBuilder CreateHostBuilder(string[] args) =>
        Host.CreateDefaultBuilder(args)
        .ConfigureAppConfiguration((context, config) =>
        {
            var configClient = new ConfigClient();
            config.AddAgileConfig(configClient);
        })
        .ConfigureWebHostDefaults(webBuilder =>
        {
            webBuilder.UseStartup<Startup>();
        });
```

使用注入的 IConfiguration 获取配置

```
[Route("[controller]/[action]")]
public class HomeController : ControllerBase
{
    private readonly IConfiguration _config;

    public HomeController(IConfiguration config)
    {
        _config = config;
    }

    [HttpGet]
    public IActionResult Index()
    {
        var value = _config["AgileKey"];

        return Ok(new { value });
    }
}
```

使用 AgileConfig

在配置页面上的配置都是字符串键值对的，那对于Json字符串应该怎么处理呢, 我们可以使用 Tuhu.Extensions.Configuration.ValueBinder.Json 扩展，通过Nuget安装即可，然后修改Startup.cs 文件

```
public void ConfigureServices(IServiceCollection services)
{
    services.ConfigureJsonValue<LogOptions>("", Configuration.GetSection("LogOptions"));
}

LogOptions：

public class LogOptions : IOptions<LogOptions>
{
      public string Level { get; set; }

      public int Count { get; set; }

      public LogOptions Value => this;
}
```

WPF 项目示例

```
public partial class App : Application
{
    public static IConfigClient ConfigClient { get; private set; }

    protected override void OnStartup(StartupEventArgs e)
    {
        base.OnStartup(e);

        var appId = "appId";
        var secret = "appSecret";
        var nodes = "http://192.168.100.65:5555";
        ConfigClient = new ConfigClient(appId, secret, nodes);

        ConfigClient.ConnectAsync().GetAwaiter();
    }
}
```

WPF 项目示例

```
Task.Run(async () =>
        {
            while (true)
            {
                await Task.Delay(5000);
                if (App.ConfigClient.Status == Agile.Config.Client.ConnectStatus.Connected)
                {
                    foreach (string key in App.ConfigClient.Data.Keys)
                    {
                        var val = App.ConfigClient[key];
                        Console.WriteLine("{0} : {1}", key, val);
                    }
                }
            }
        });
```

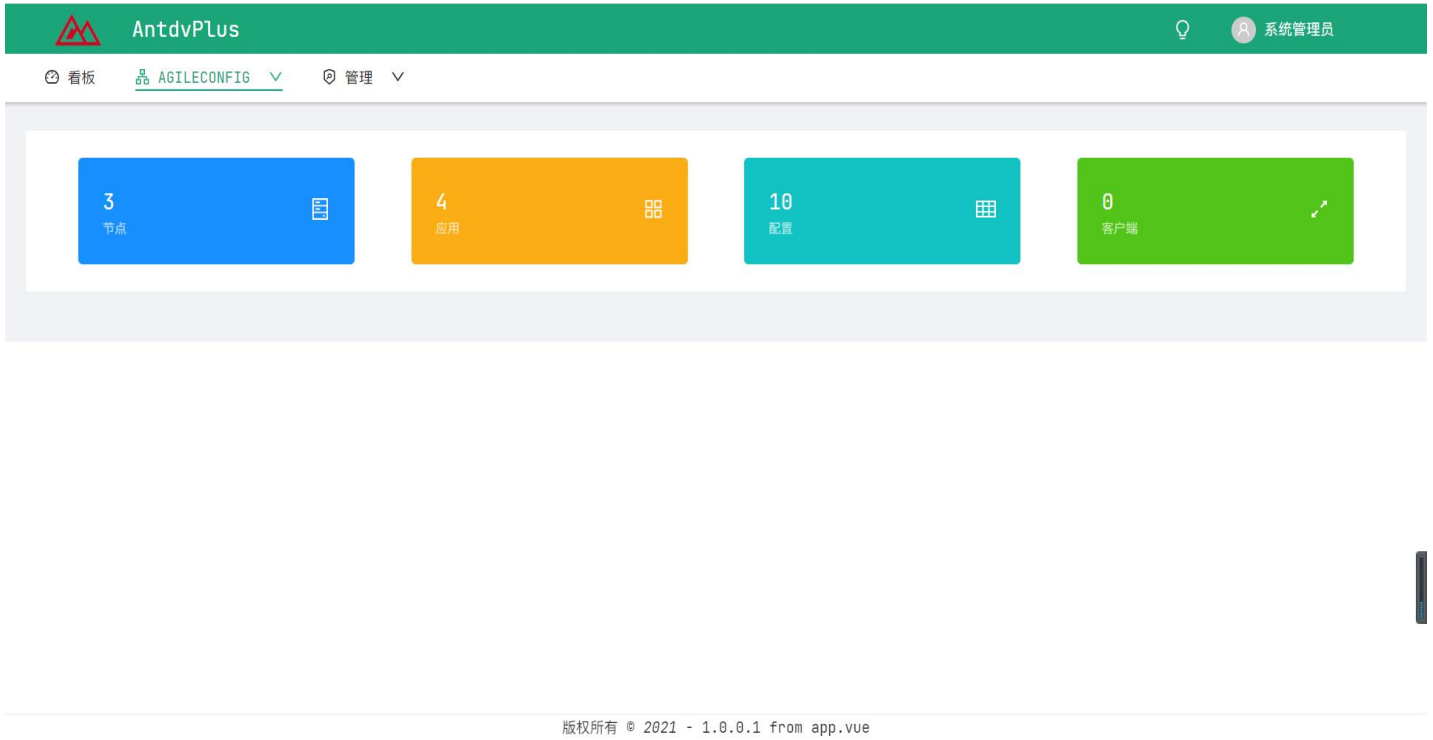
总结

AgileConfig 是使用 .net core 开发的配置组件，部署和使用起来都很简单，但是目前还有一些不足，比如多账号权限管理，和多环境支持，一般是开发，灰度和正式，不过没有关系，项目都是开源的，大家感兴趣的一起建设，完善不足的功能，现在.NET 社区是越来越好了，如果对大家有帮助的，可以支持一下！

06 定制化

前端定制

采用了AntdvPlus作为控制台管理前端，配合 JWT 认证机制，实现作者的 React UI 前端替换。



后端定制


```
▼{success: true, data: [{address: "http://192.168.100.20:5010", remark: "控制台节点", status: 1,…}]}
  ▶data: [{address: "http://192.168.100.20:5010", remark: "控制台节点", status: 1,…}]
   success: true
```



```
▼{message: "", code: 200, isSuccess: true,…}
   code: 200
  ▼data: {success: true, data: [{address: "http://192.168.100.20:5010", remark: "控制台节点", status: 1,…}]}
    ▶data: [{address: "http://192.168.100.20:5010", remark: "控制台节点", status: 1,…}]
      success: true
   isSuccess: true
   message: ""
```

更多场景

- 无锡机场安检效能客户端
- 无锡机场安检效能大屏
- AgileConfig.Client 的 Java 版本

Thanks