进入公司正好赶上公司的云组网项目重构。云组网是公司为电信部署的路由器，协同器等设备的开发的一套设备管理系统。

**系统架构：**由于一开始原来的云组网由于涉及的功能模块少，为了快速完成功能需求。采用的是单体架构用，因为单体架构比较容易部署、测试所有模块全部放在一起， 在项目的初期，单体应用可以很好地运行。然而，随着需求的不断增加， 云组网的需求越来越多，越来越多的人加入开发团队，代码库也在飞速地膨胀。慢慢地，单体应用变得越来越臃肿，可维护性、灵活性逐渐降低，维护成本越来越高。代码之间的耦合也十分高，每次修改都经常牵一发而动全身。像我们这样的新人刚进公司很难快速理解代码，造成公司培养成本增加，过度依赖老人维护开发老的系统，不利于公司健康发展。而且配置文件过于分散不易于管理，每次对配置文件的修改都需要对程序重新启动。

重构基于微服务架构。讲项目进行拆分若干个服务， 每个服务为独立的业务开发，一个微服务只关注某个特定的功能。服务注册和发现框架采用了阿里巴巴的开源项目NACOS，它是针对微服务架构中的服务发现、配置管理、服务治理的综合性解决方案。

整个项目组分为前端使用veu框架，测试，后端分为：前台：主要负责业务功能的实现java语言基于springboot框架；中台负责公共工具的开发以及一些中间件的部署；后台负责采集设备的数据以及接收前台下发的业务层任务对设备的执行，基于go语言。

**负责模块**：我跟着另外两个同事在 前台进行业务模块的开发，主要负责资源和工单模块的开发；工单模块主要是用于任务的制定、保存、部署下发、设备状态展示；资源模块用于对设备的管理模块，例如增加设备，删除设备对ip地址的分配和占用，新增设备需要对其分配的IP地址判断是否ip重复等等，记录设备节点所属省市 ，保存设备对用ip地址，物理端口，逻辑端口 等等展现给用户，使用户知道自己的设备，以及设备配置，为工单模块，工单创建提供资源支持。

**遇到的问题**：在重构的同时，每个员工都会被分配改case任务也就是现网交付给客户出现的bug。一次线上故障，数据库查询时间过长，导致前端页面频频报错，结果不仅该服务的访问受到了影响，其他服务的访问的流畅度也下降了。打印当前页的业务日志拷贝sql。由于sql很长不是很方便直接分析，然后立即对sql语句进行分析通过explain（key）命令查看索引是否生效，发现索引并没有生效。对过滤字段排查发现，开发人员在没有测试sql，导致索引没有遵循最左前缀原则。一开始数据少用户没有感觉，随着数据库内容的不断只能多，查询等待时间明显增长。

一次项目在部署到测试环境，服务器频繁报警CPU过高，影响性能。排查半天也不知道什么原因，于是去问我指导人，他让我自己去服务器看一下哪个程序占用cpu大。自己不了解JVM于是我边上网搜索边查看日志。本想通过TOP命令获取到占用CPU较高的线程，再去看看对应线程的堆栈信息。找到CPU占都不超过1%。打印GC的信息出来吧。通过jstat -gcutil  pid 查看虚拟机的堆内存信息发现可怕的是FGC占用的时间高达1000多s发现频繁的进行fullGC，因为这十几分钟java应用进程是STW。说明是内存溢出导致的系统变慢。由于一般不提倡修改JVM参数，优先优化代码。怀疑有人在代码中使用System.gc();原来是为了要统计数据，先把数据全部查出来，然后遍历数据分类统计。 又意识到这样会有生成很多大对象（多的话会有几十万数据放在这个List对象），所以就手动做垃圾回收。后来用sql直接统计出来数据，那样就不会生成大对象了。