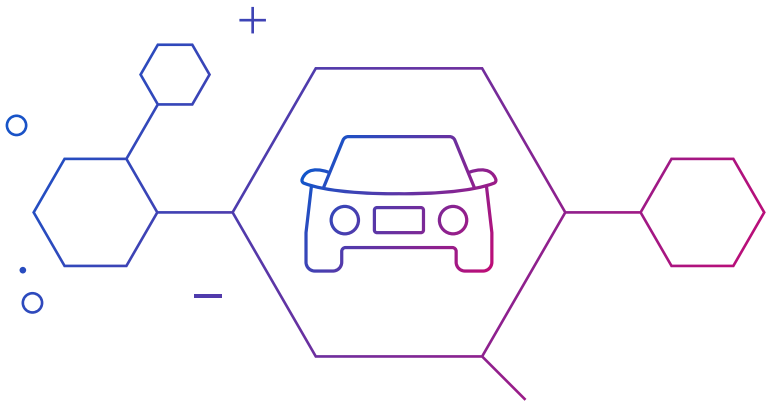
亚马逊云科技

车联网单元架构白皮书

目 录

CONTENTS



免责声明

本白皮书陈述了亚马逊云科技在封面页所示日期的有关服务产品及实践，该等信息可能变化且我们不会另行通知。客户对于本白皮书的信息以及亚马逊云科技的产品或服务应自己做出独立的判断，该等内容都是“依现状”提供，不包含任何明示或者暗示的保证。本白皮书并没有创设来自亚马逊云科技、北京光环新网科技股份有限公司（“光环新网”）、宁夏西云数据科技有限公司（“西云数据”）、或其各自的关联方、提供方或许可方的任何保证、陈述、合同性承诺、条件或者担保。亚马逊云科技、光环新网、西云数据对其各自的客户的义务和责任均由适用的客户协议管辖。本白皮书不是亚马逊云科技、光环新网、西云数据和其各自的客户之间任何协议的组成部分，也不构成对任何协议的修改。

新能源汽车车联网面临的挑战

中国新能源汽车市场已稳居全球首位，2024 年基于中汽协的数据显示其渗透率已达 48.9%，政策驱动（双积分、购置税减免）与技术突破（固态电池、800V 平台）推动产业高速发展。发展趋势呈现 " 电动化 + 智能化 + 生态化 " 三位一体转型，根据乘联分会发布《2024 年 12 月汽车智能网联洞察报告》指出，2024 年新能源乘用车 L2 级及以上的辅助驾驶功能装车率将突破 60%。车联网技术作为战略核心，通过 5G-V2X、OTA 升级、云端孪生等技术实现数据闭环，重构用户体验、商业模式（订阅制）和产业边界，成为车企从 " 硬件制造商 " 向 " 移动服务商 " 跃迁的关键支点。

随着车联网覆盖用户的规模越来越大，其稳定性已成为用户体验的隐形战场，在新能源汽车 " 软件定义汽车 " 的变革浪潮中，车联网平台稳定性至关重要。

全生命周期用户运营需 7x24 小时服务在线，故障 1 小时可能导致数百万元损失；	同时应对高峰时段百万车辆并发请求，需云平台无感扩容；	毫秒级响应的车联控制以及智能座舱的语音交互提升用户体验的同时需要更高的稳定性；

传统的车联网平台架构在面对海量设备接入、数据激增、带宽受限以及时延敏感等挑战时，逐渐显露出其局限性。智能车辆正在以前所未有的速度产生海量数据。特别是在 " 全量数据上云 " 的传统处理模式下，单一区域部署难以满足当前车联网系统对实时性和效率的严苛要求。同时面对极致车控体验、云游戏、大模型等新兴智能化应用场景带来的挑战，传统车联网架构也已难以为继。

为此我们提出单元架构的理念，单元架构具备故障隔离、易扩展、快速部署以及就近服务的特性，可以帮助高速发展的车企应对上述挑战。单元架构将计算能力前移，在物理位置上更贴近用户端和数据产生源，不仅是对传统车联网平台的重要补充，更是一次架构层面的革新性突破。通过就近提供服务和数据处理能力，单元化设计有效解决了传统架构中的痛点问题。

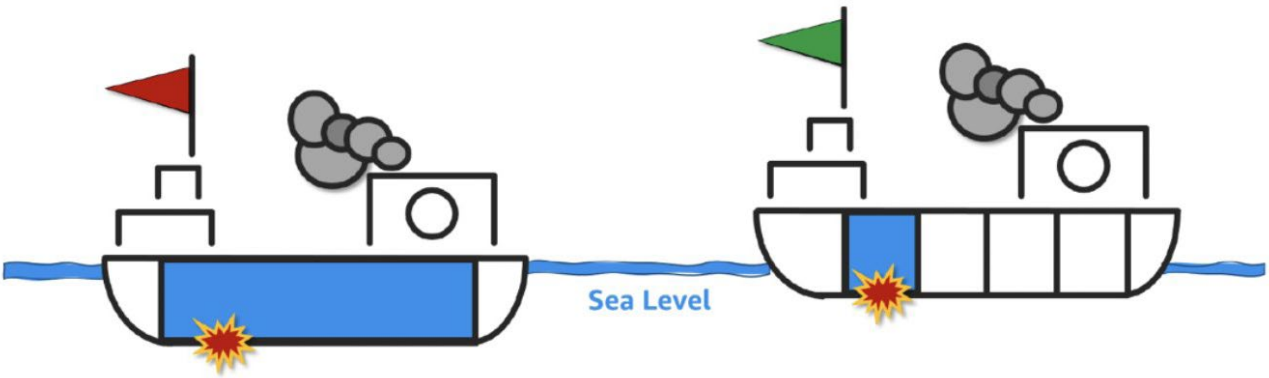
值得骄傲的是，中国汽车厂商在这场技术变革中展现出了非凡的创新勇气和前瞻眼光，已经有部分厂商率先将单元架构设计理念融入车联网系统建设。这不仅体现了中国汽车产业在智能网联领域的技术实力，更展现了引领全球车联网技术发展的雄心。通过创新性地应用单元架构，中国车企正在为全球智能汽车产业开创一条全新的发展道路。

后续将阐述单元架构的定义、组成，单元架构在亚马逊科技上的构建方法以及单元架构在车企中的落地实践。

单元架构
原理与优势

02

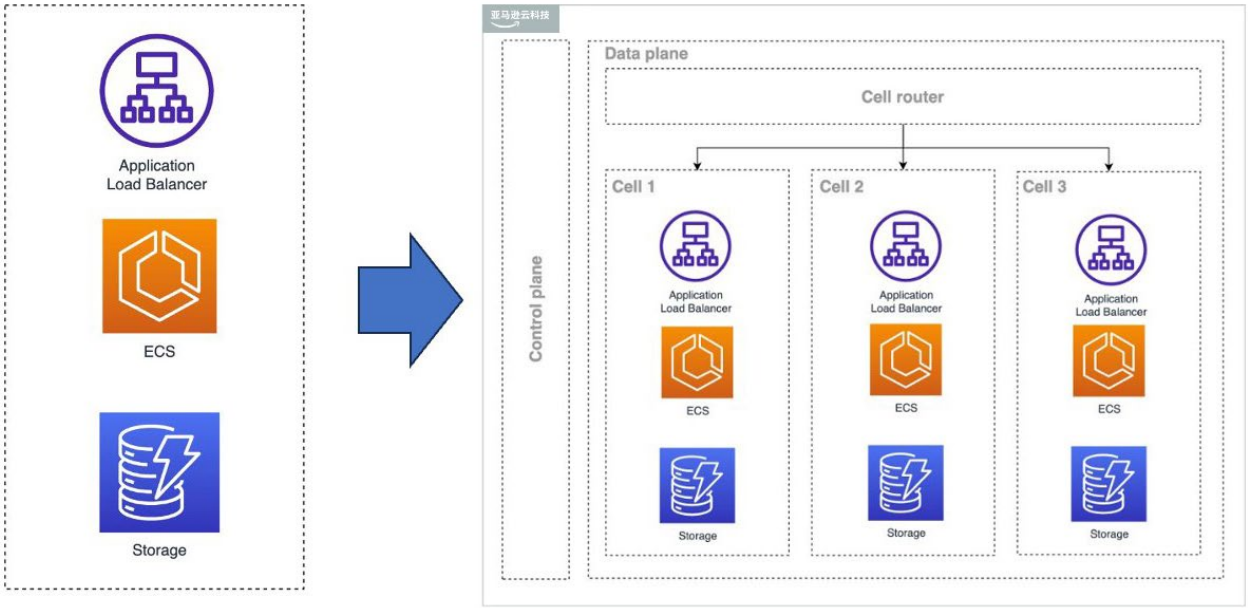
2.1 单元架构的核心概念与设计理念



单元架构 (Cell-Based Architecture) 源自船舶隔舱的概念，其中垂直隔墙将船舶内部细分为独立的水密隔间。隔舱可减少损坏时海水淹没的程度，并为船梁提供额外的刚性。在复杂的 IT 系统中，通常会采用这种模式以实现故障隔离。故障隔离边界将工作负载内故障的影响限制在有限数量的组件内。边界之外的组件不受故障影响。

单元是指一个能完成所有业务操作的集合，即单元化应用服务层的部署单元，是一个独立完整的业务处理闭环，在这个集合中包含了所有业务所需的所有服务以及分配给这个单元的数据。**单元架构**就是将单元作为部署的基本单位，在业务需要覆盖的范围内部署多个单元，任一单元均部署系统所需的全部应用，数据则是全量数据按照某种维度划分后的一部分。基于单元架构部署相同工作负载的多个隔离单位，其中每个单位称为一个单元。每个单元都是独立的，不与其他单元共享状态，并处理总体工作负载请求的子集。这减少了故障（例如错误的软件更新）对单个单元及其正在处理的请求的潜在影响。如果一个工作负载使用 10 个单元来服务 100 个请求，那么当一个单元发生故障时，90% 的请求将不会受到故障的影响。

通过使用多个隔离边界，可以限制故障对工作负载的影响。在配置新客户、租户或应用工作负载更改时，您可以逐个隔离区的执行操作。这样，当发生故障时，就可以减少受影响的客户或资源的数量。在亚马逊云科技上，客户可以使用多个可用区（Availability Zone）和区域（Region）来进行故障隔离，但相同的概念也可以扩展到车联网系统架构中。



一个典型的工作负载　　基于单元架构的工作负载

为此可以将一个典型的三层架构进行改造，增加路由层和单元的控制平面（管理系统）后，就构成新的基于单元架构的系统。单元架构具有以下组件：

单元路由层

单元路由层负责流量的路由层根据单元的映射算法控制流量分发到各个单元的。其路由机制可使用分区映射算法将分区键映射到相应的单元。也可结合业务逻辑进行分配。

控制平面

（单元的管理系统）单元的管理系统负责各个单元的创建、部署以及用户在单元间的迁移。

单元

单元应用系统和通常的系统部署类似，运行完整的工作负载，具有独立运行所需的一切。

2.2 单元架构帮助车联网应对挑战

针对车联网面临的可用性、可扩展性和低延迟的要求，单元架构的如下设计特点可以帮助车联网应对挑战：

- **更小的爆炸半径**

将服务分散到多个单元可以减少影响范围。单元代表隔舱，可以遏制许多常见故障，例如部署失败、行为不当的客户端、数据损坏和操作错误。当正确的隔离后，服务中断不太可能跨越多个单元。

- **更高的可用性 = 更高的平均无故障时间 + 缩短平均恢复时间**

单元架构不仅仅减少了故障影响，它甚至可能减少故障本身，单元架构的最大容量限制以及可以经常性的进行测试行为，减少了因每天不断变化所导致的动态风险。

在日常运营中，客户分布在各个单元中可以就地识别问题，例如您的客户分布在多个单元中，如 10 个单元，因此现在每个单元中都有 10% 的客户，就可以更好地管理系统变更并控制某些单元中的代码或流量峰值等故障影响范围。对于新版本应用程序的部署，如果每次发布只应用于少量单元，则可以在识别出故障时进行回滚，只影响少量客户。由于其他单元保持稳定且不受影响，从而增加应用程序的整体平均无故障时间。

单元也更容易恢复，因为它们限制了需要分析和进行问题诊断以及紧急代码和配置部署的主机数量。单元带来的大小和规模的可预测性也使得在发生故障时恢复更加可预测。

- **更好地控制部署和回滚的影响**

单元的分阶段部署可以减少有问题的部署的影响范围，使用单元可以使部署和回滚更顺畅。可以一次将更改部署到一个单元，从而最大限度地减少代码和部署问题带来的影响。

上述三点可以帮助车联网缩小局部故障后的对终端用户的影响范围，提高总体可用性。

- **应对高峰时刻容量扩容，横向扩展优于纵向扩展**

单元架构将系统的扩展方法从纵向扩展变为横向扩展。每个单元是一个完全独立的容量单位，可以定义和管理每个单元的容量，可以添加更多的单元，每个单元都包含已知容量，从而使系统具有固有的可扩展性。这种设计更改不会改变您的服务的客户体验，客户可以像现在一样继续访问服务。

通过添加更多单元以进行扩展并平均分配工作负载来处理增长。这避免了由于扩展 (通过增加数据库、服务器或子系统等系统组件的大小来适应增长) 而导致的资源限制。随着需求的增长，可以添加更多的单元，每个单元都包含已知容量，从而使系统具有固有的可扩展性。

- **便于测试**

测试分布式系统是一项具有挑战性的任务，并且随着系统的增长而变得更加艰巨。

使用限制组件的最大尺寸，测试也变得更容易，可以对这样组件进行压力测试并突破其极限点，以了解其安全运营冗余度。如果系统的大部分复杂性和风险都经过压力测试（而且应该如此），则测试覆盖率和置信度水平会显着提高。

由于成本原因，大规模服务定期模拟所有租户的全部工作负载是不切实际的，但模拟单元可以容纳的最大工作负载是合理的。

上述这两点可以帮助车联网精准、快速的应对业务扩展，服务更多的终端用户。

- **单元的就近部署**

车联网完成单元化改造后，可以根据业务的发展选择离客户最近的地域进行快速部署，避免了中心化部署离某些客户距离较远的问题，可以为客户提供低延迟的访问、提升终端用户体验。同时还可以根据业务和客户的发展对单元的容量进行调整。

2.3 单元架构与双活 / 多活容灾的区别

单元架构经常和容灾中的双活 / 多活策略进行对比，二者是不同时代、不同行业的技术产物，应对不同的业务挑战。单元架构和 双活 / 多活架构都是为了提升系统的高可用性和可靠性而设计的分布式架构，但它们的目标、应用场景和设计理念有所不同。

单元架构是一种将系统按业务功能划分为相对独立单元的架构模式，每个单元能够独立提供完整的业务服务。而双活 / 多活容灾则是在多个地理位置同时部署功能相同的活跃系统副本，以提供容灾能力和服务连续性。单元架构按照业务功能或领域进行划分，强调的是功能的自治性。双活 / 多活容灾则主要按地理位置进行划分，重点是冗余。在单元架构中，数据通常按业务功能进行分片和隔离，减少单元间的数据依赖。而双活 / 多活容灾更强调跨地域的数据同步与一致性，需要解决分布式数据复制和冲突处理问题。二者并不是对立的，可以根据业务需要进行融合。

详见附录说明。

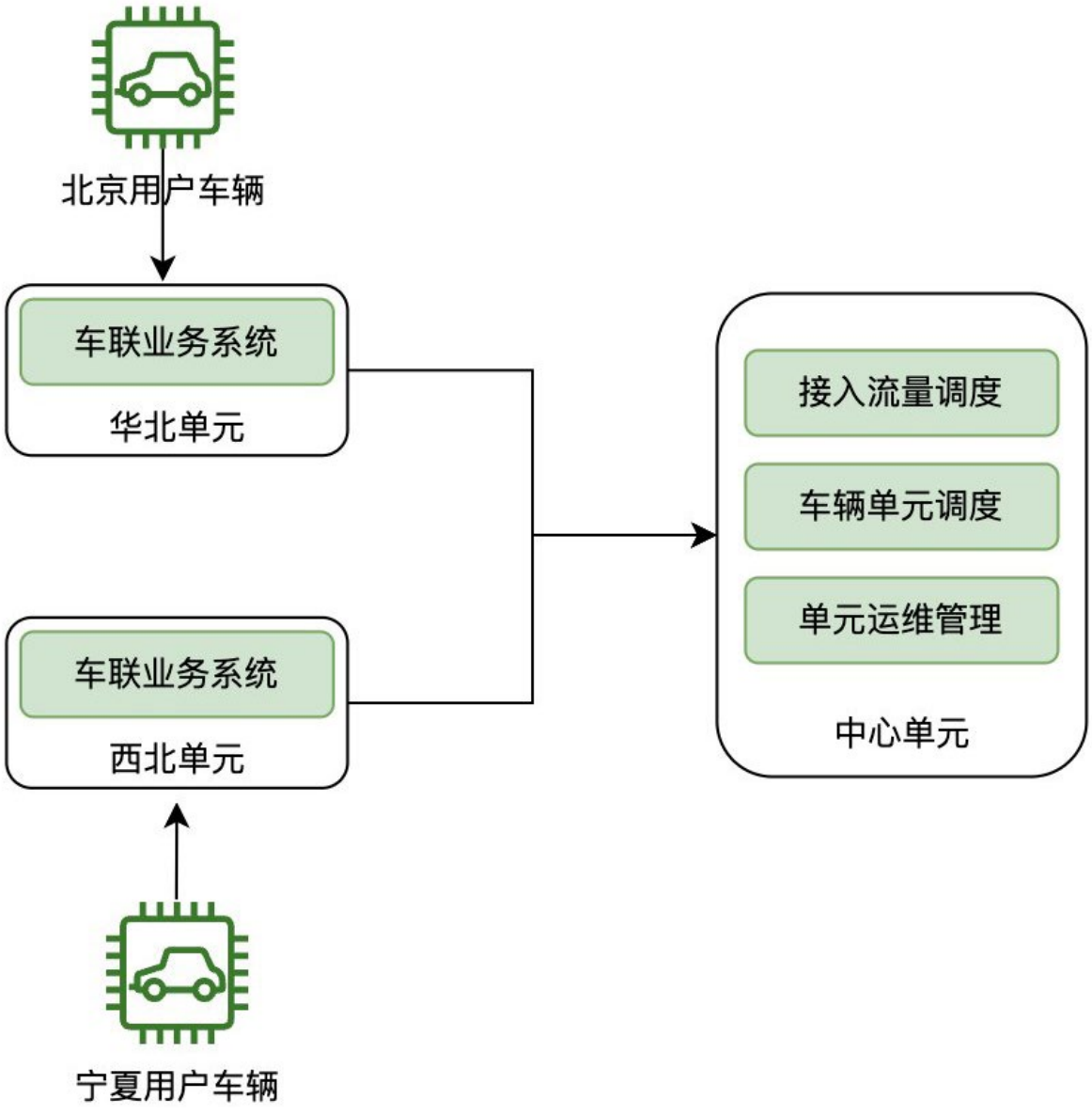
单元架构在车联网中的应用概述

03

3.1 车联网的单元化部署

单元化车联网是一种创新的网络架构，融合了边缘计算和单元化部署的先进理念。它将计算和存储资源部署在靠近数据源的边缘节点上，如车辆和路侧单元，实现低延迟、高带宽和高可靠性的车联网服务。通过对这些资源进行单元化部署，实现了车联网业务的故障隔离和灵活扩展，提高了系统的稳定性、可扩展性和适应性。

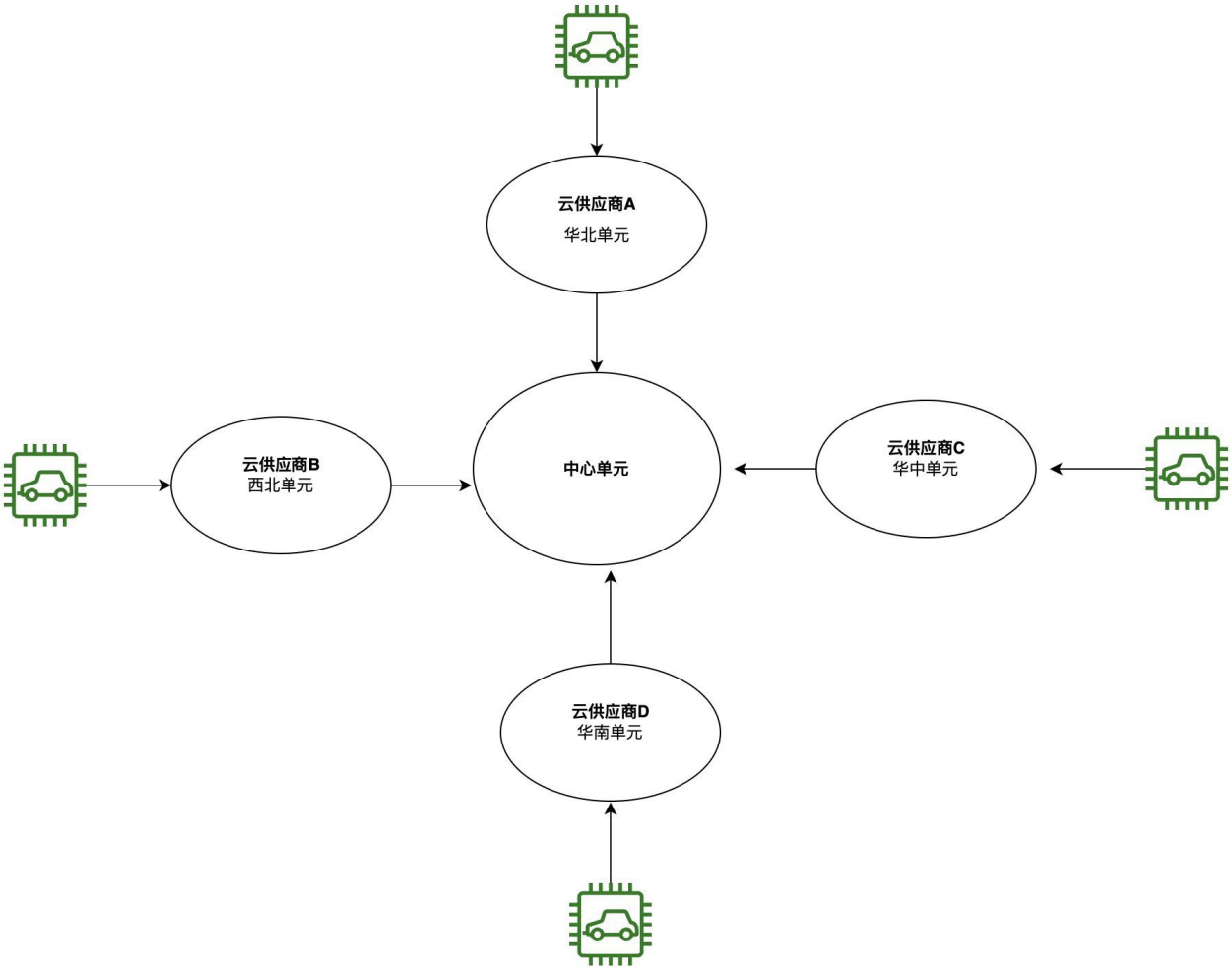
典型的单元化车联网架构由两个核心组件构成：**中心单元和区域业务单元**。中心单元包括负责流量接入调度的路由层和管理单元部署及车辆单元调度的单元管理层（详见第四章以及第五章）；
区域业务单元则部署在各个地理区域，处理本地化的车联网业务（详见第六章）。



单元化的车联网具有如下优势：

- **低时延，稳定的业务通信；**
- **满足业务全新的场景化需求。大模型上车，车端云游戏；**
- **故障隔离，某个单元故障只影响该单元内部署的车辆，缩小了故障的影响范围。**

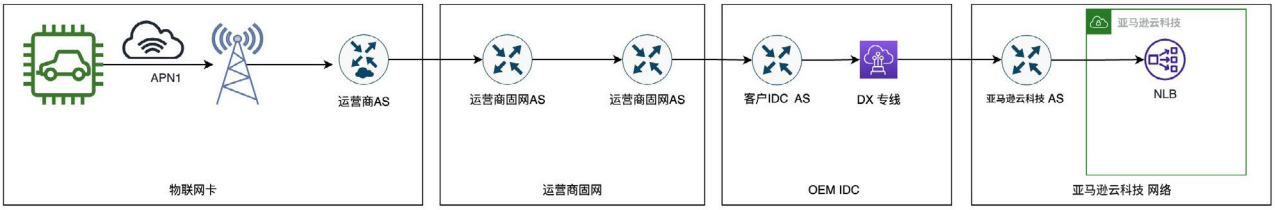
同时单元化部署的车联网，可以充分利用国内各个云供应商的资源，利用其区域覆盖优化单元化车联网部署在更靠近用户的地方。



3.2 车联网的单元化部署的网络基础

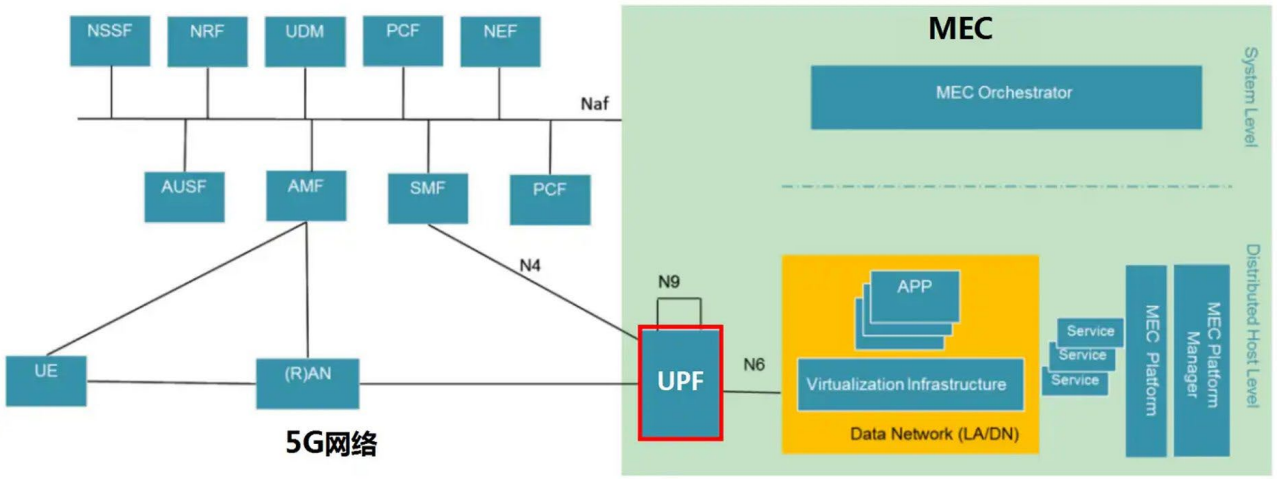
车联网单元架构引入了一种创新的连接模式，其中车辆根据特定属性与区域单元建立连接，这与传统依赖 4G 网络的车联网模式有着本质区别。这种先进架构的全面实现，很大程度上依赖于 5G 网络的广泛普及。

回顾上一代基于 4G 的车联网，其网络链路冗长，全国各地的数据流量需要汇聚至统一的物联网出口网关，导致某些地区的网络延迟甚至超过 100 毫秒。这种高延迟不仅无法充分发挥单元化车联网的优势，还严重制约了实时应用的发展。此外，4G 网络最后一公里通常需要通过专线连接云端，随着智能网联车辆数量的急剧增长，这种架构在稳定性和可扩展性方面面临巨大挑战。

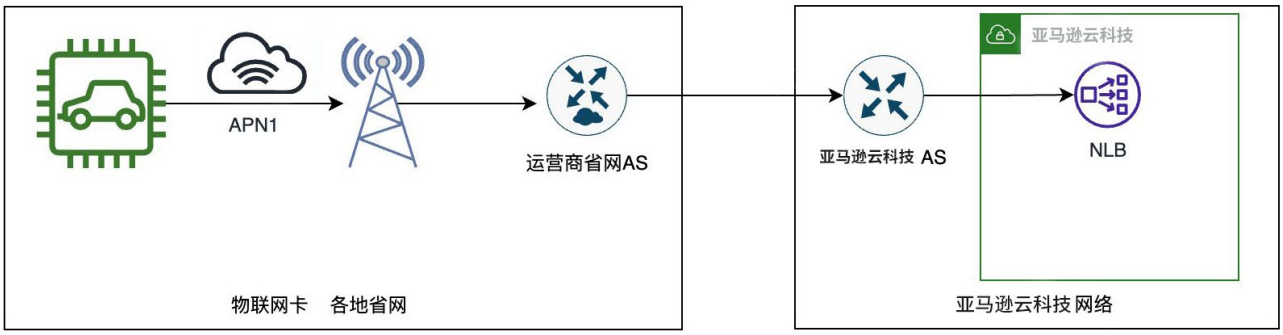


5G 核心网最关键的网元：UPF（User Plane Function，用户面功能），是连接 5G 核心网和车联网各个单元的纽带，可提供数据分流及流量统计等功能。互联网厂家的边缘计算平台需要和运营商的 UPF 对接，把运营商的网络作为传输管道。

更多细节请见：5G 行业虚拟专网网络架构 - 中国信息通信研究院、5G+MEC+V2X 车联网解决方案白皮书



单元化车联网使用 5G 线路，车辆和云端直接通信，网络路径更短（如图），可以让车辆接入就近的单元（如图），获得更极致的体验。

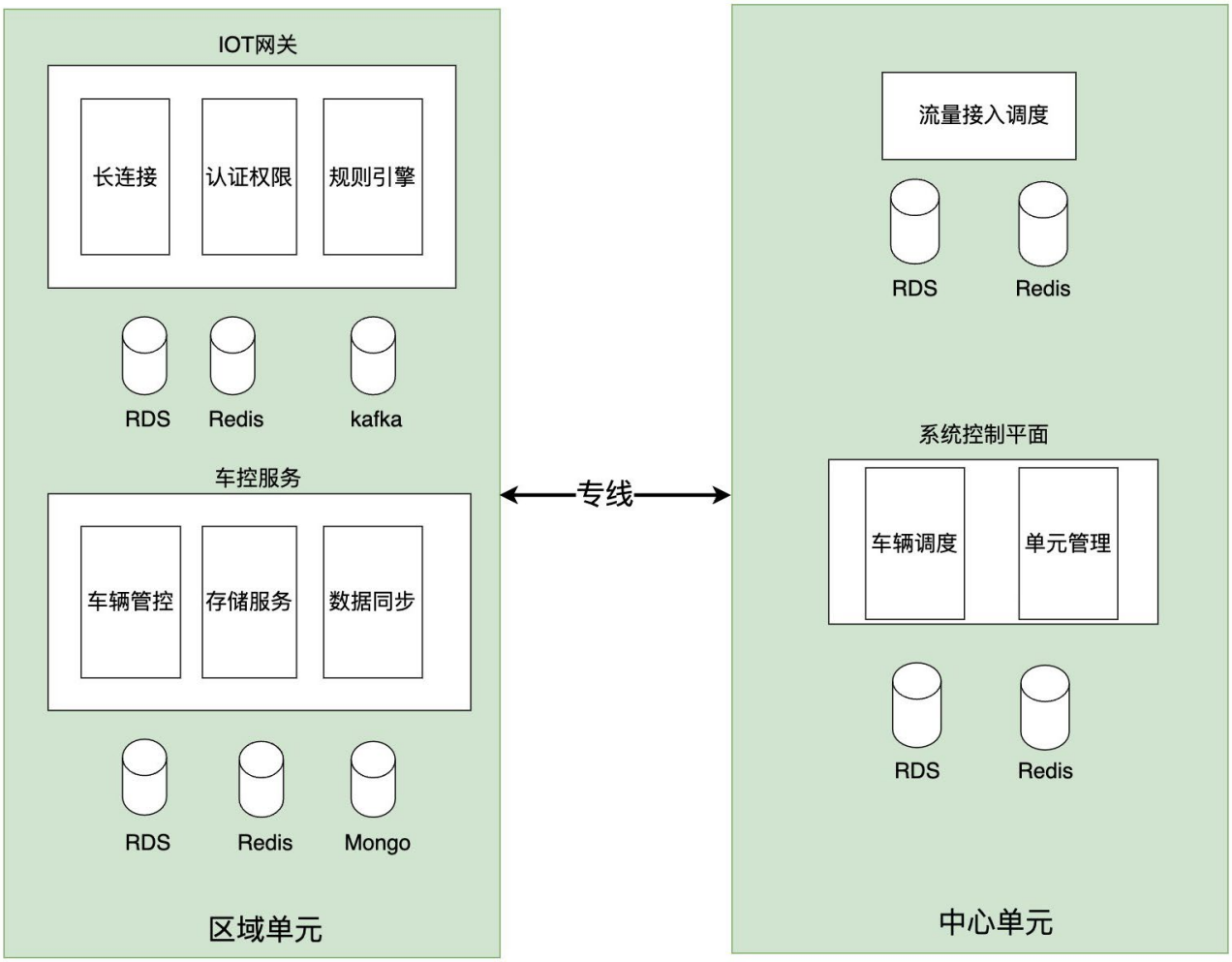


3.3 单元化车联网架构概览

一个典型的单元化架构部署的车联网由**中心单元**和**区域业务单元**组成。

中心单元包括负责流量接入调度的路由层和系统控制面（管理单元部署及车辆单元调度的单元管理层（详见第四章以及第五章节）；区域业务单元则部署在各个地理区域，处理本地化的车联网业务，其业务系统主要包括 IOT 网关和车控服务两部分（详见第六章节）。

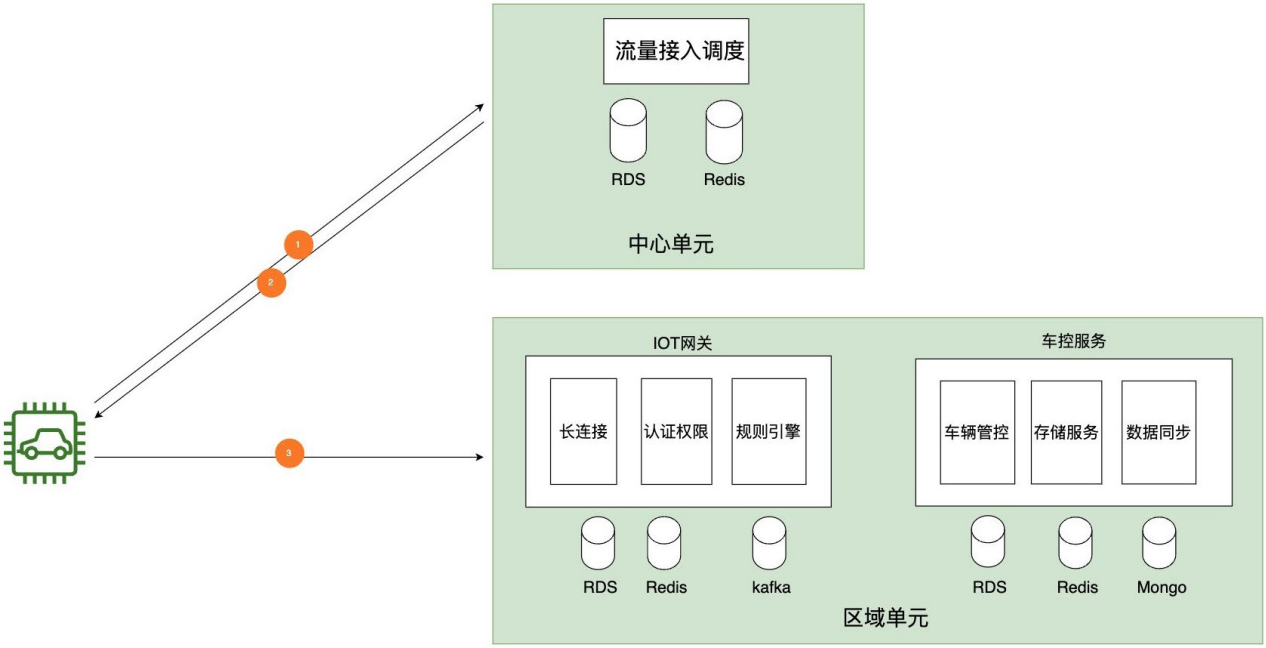
以亚马逊云科技为例，可以流量接入调度的路由层和系统控制平面部署在北京 region，并同时在北京 region 和宁夏 region 各部署车联网区域单元。



3.4 单元化车联网核心业务工作流示例

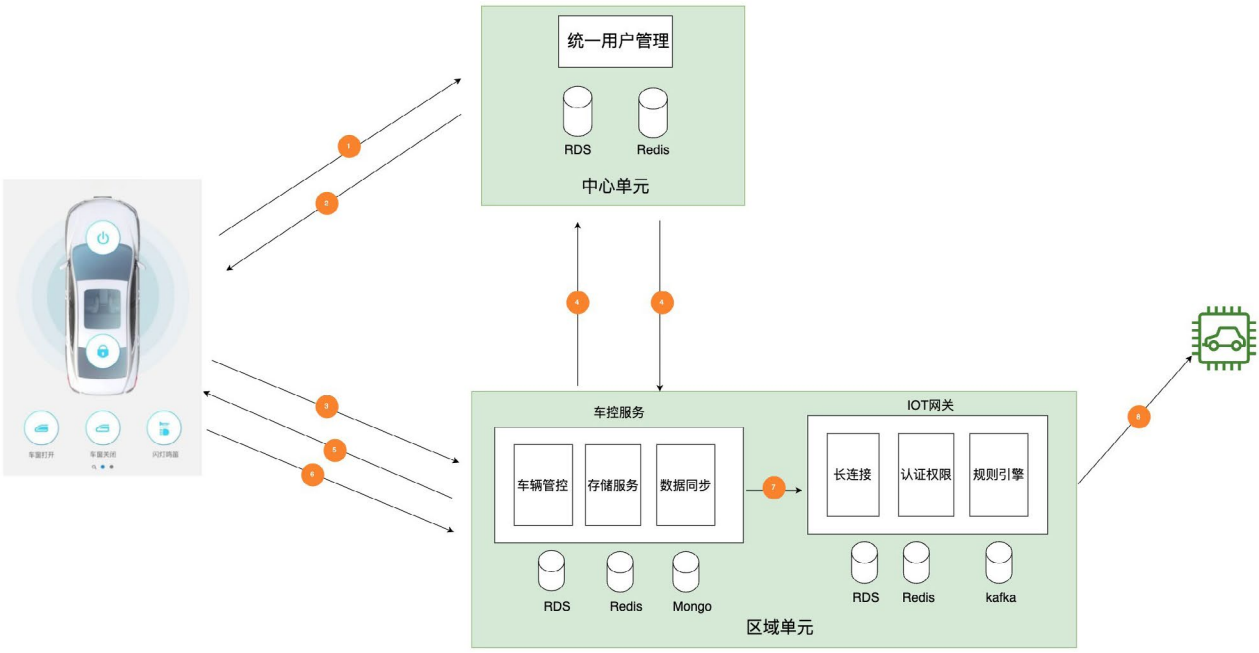
单元化车联网架构是以车辆为纬度进行车联网业务的单元化切分设计，旨在实现车联网业务的高效单元化部署。然而，值得注意的是，车联网系统中还包含其他关键组件，如用户管理系统和大数据业务系统，这些系统目前还采用了中心化的设计方法，为了保持本白皮书的重点和清晰度，我们将主要聚焦于单元化车联网架构的核心概念和实现。尽管如此，在阐述业务流程时，我们可能会涉及到与这些系统的交互，以便读者能够更全面地理解整个系统的运作。

3.4.1 车辆连接的工作流



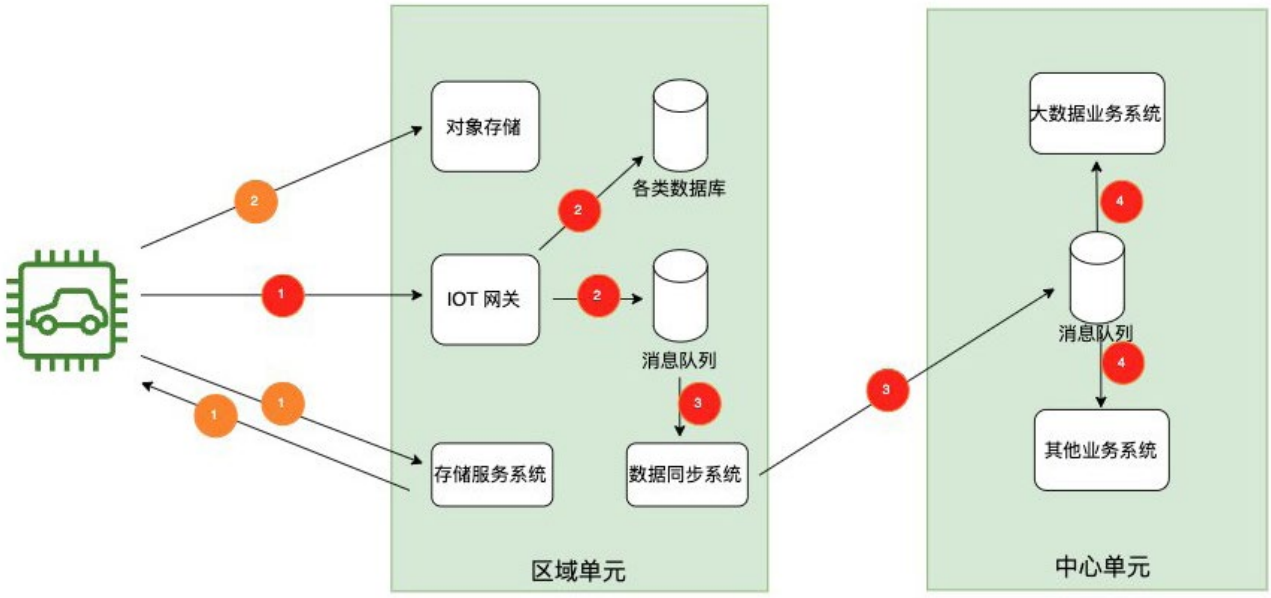
1. **请求到达：**车辆发起 HTTP(S) 请求到达中心单元流量接入调度平台
   解析请求：解析请求头或 URL 获取车辆 Vin 标识
   查询单元数据中心：通过单元数据中心获车辆所属的单元间关系
   路由决策：决定是否需要将请求重定向到目标单元内的接入网关
2. **响应返回：**调度平台获取正确的单元接入网关地址，然后传递回车端
3. **重新请求：**车辆请求接入对应区域单元的接入网关，完成接入

3.4.2 远程指令下发流程



1. APP 用户登陆认证
2. 返回用户权限凭证信息
3. 使用凭证信息请求车控服务
4. 和中心元服务交互，验证凭证信息，并本地缓存
5. 返回车辆的信息
6. APP 端 https 请求车辆远控指令
7. 车控服务转发指令消息到 IoT 网关
8. IoT 网关通过 mqtt 下发指令消息到用户车辆

3.4.3 车辆数据上行流程



非 IoT 消息

1. 获取边缘对象存储信息 (配置和权限)
2. 上传数据到对象存储

IoT 消息

1. IoT 消息通过 MQTT 上行到 IoT 网关
2. IoT 网关利用规则引擎路由消息到不同的数据库，提供给对应业务
3. 数据同步系统跨云同步消息到中心单元的消息对列
4. 中心单元业务大数据系统汇总处理各区域单元车辆数据，提供业务方使用

因为数据上行依赖区域单元的 IOT 网关系统，数据不能直接上传到中心单元的大数据业务系统中。

流量调度层（路由层）的功能说明和实现

04

单元架构的路由层根据单元的映射算法控制流量分发到各个单元，完成流量接入调度的功能。其路由机制可使用分区映射算法将分区键映射到相应的单元。也可结合业务逻辑进行分配，分配的方法需要确保：

请求被均匀分布到各个单元	相同的分区键总是路由到相同的单元	路由策略可以快速计算，无需复杂的查找

对于车联网单元架构，目前建议根据车辆所在地域（主要使用地域）进行单元归属的划分，未来可根据用户等级属性、或车型等级属性建立 VIP 单元。

路由层的设计要求：

- 尽可能简单
- 在单元之间实现请求分发隔离
- 对客户端抽象底层单元实现和复杂性
- 快速且可靠
- 即使一个单元不可达，也能继续正常操作其他单元
- 遵循静态稳定性设计原则，边缘的单元站点的运行不应对路由层有依赖

路由层是整个系统的关键组件，需要特别注意其韧性和扩展性：

- 避免单点故障，应根据要求实现跨可用区甚至跨 region 部署，需要实现实现足够的冗余和自动故障转移
- 路由层应能处理所有单元的总流量，实现自动扩展以应对流量峰值，具备熔断和超时机制，应对突发流量
- 监控和可观测性，实时监控路由层健康状况，设置适当的警报和自动恢复机制
- 在路由层实现适当的缓存，确保缓存一致性和过期策略
- 实现安全的配置更新机制，考虑使用金丝雀发布进行路由更改

对于路由层的实现，通常有两种方法，**基于 DNS 层路由实现**、**基于网关和存储服务实现**，网关可以采用托管的 API Gateway 或基于 EC2、EKS、lambda 等计算服务搭建，映射关系可存储在 NoSQL 数据库或关系型数据库中，具体根据业务规模和客户现有环境选择。

4.1 基于 DNS 层路由实现

基于 DNS 层路由实现，以使用 Amazon Route 53 为例，Amazon Route 53 可以作为 DNS 层路由器，基于地理位置或延迟的路由策略并结合加权路由以控制流量分配，并利用健康检查功能自动将流量从不健康的单元转移。

DNS 路由本质上是无状态的，每次 DNS 查询都是独立的操作，其路由决策基于预定义的规则和配置，而不依赖于之前的请求历史，不需要在路由层维护会话状态，简化了设计和实现，DNS 系统本身具有分布式特性，通常部署在多个地域，例如使用 Amazon Route 53 等服务可提供 100% 的可用性 SLA，即使部分 DNS 服务器不可用，整体路由功能仍能继续工作。

但 DNS 路由通常提供相对粗粒度的流量控制，难以实现精确的流量百分比分配，且路由变更需要等待 DNS 缓存过期才能完全生效。并且虽然 Route 53 等服务提供健康检查功能，但检查频率和响应速度有限，无法立即检测到所有类型的故障，特别是间歇性问题，其健康检查通常基于简单的探测机制，难以模拟复杂的用户交互场。

基于 DNS 层路由的调度，不太适合车联网单元架构的调度，因为

车辆网络连接通常不稳定，可能频繁切换网络（4G/5G/Wi-Fi），DNS 缓存在网络切换时可能失效，导致频繁重新解析，车辆在移动过程中可能穿越不同网络区域，影响 DNS 解析结果。

车联网应用（如远程控制、实时导航、安全警报）对延迟高度敏感， DNS 解析引入的额外延迟可能不可接受，初始 DNS 查询延迟可能影响紧急情况下的响应时间。

车联网系统对可靠性有极高要求，特别是安全相关功能，DNS 解析失败可能导致关键服务不可用，DNS 缓存过期或污染可能导致连接到错误或不可用的服务端点。

车辆位置快速变化，基于地理位置的 DNS 路由可能无法及时响应，车辆可能在短时间内跨越多个服务区域边界，DNS TTL 设置难以平衡及时更新与缓存效。

虽然 DNS 路由在互联网基础设施中扮演着重要角色，但在专业的车联网系统中，DNS 路由存在很多不足，为此提出基于 VIN 码的路由机制，以提供更高的精确性、灵活性、安全性和性能。说明如下。

4.2 基于 VIN 码的路由机制

基于 VIN 码的路由机制巧妙地利用了 VIN 码的唯一性和丰富的信息内容，通过用户自定义路由规则，将车辆精确地分配到不同的区域单元。这种方法不仅保证了系统的高效运作，还为未来的扩展和优化提供了广阔空间，实现了高效、灵活和可扩展系统。

路由服务作为系统的中枢，维护着至关重要的全局路由表，动态管理 VIN 码与单元之间的对应关系。为了进一步提升系统响应速度和降低中心服务器的负载，其设计时可增加路由信息缓存机制。这种设计不仅确保了整体负载的均衡分布，还通过考虑车型特性、地理位置等多维度因素，实现了更为精细和可控的资源分配策略。

在实际运营过程中，单元切换是保持系统动态平衡和高效运行的关键机制。这种切换主要发生在以下几种典型场景：

负载均衡：

当系统检测到单元间负载出现显著不均衡时，会自动触发重新平衡机制。这个过程涉及将部分 VIN 码重新路由，以优化资源利用，确保每个单元都能高效运作。

单元变动：

在新单元加入或现有单元退出的情况下，系统需要重新分配受影响的 VIN 码。这种动态调整能力保证了系统的可扩展性和灵活性。

故障处理：

当发生单元故障或网络中断等紧急情况时，相关 VIN 码会迅速自动切换到预设的备用单元。这种快速响应机制确保了服务的连续性和可靠性。

系统维护：

当发生单元故障或网络中断等紧急情况时，相关 VIN 码会迅速自动切换到预设的备用单元。这种快速响应机制确保了服务的连续性和可靠性。

用户体验优化：

基于用户位置的显著变化或使用模式的重大改变，系统可能决定将某些 VIN 码切换到更适合的单元。这种基于用户需求的动态调整，旨在提供更加个性化和高质量的服务体验。

基于 VIN 码的路由机制具有如下优势：

1. 粒度和精确性

DNS 路由主要基于域名解析，难以实现对单个车辆的精确路由。相比之下，VIN 码路由能够精确到每一辆车，提供更细粒度的控制和管理。这种精确性对于车联网系统的个性化服务和精细化管理至关重要。

2. 灵活性限制

DNS 系统的更新和传播机制相对较慢，不适合频繁变更的动态环境。车联网需要能够快速响应车辆位置变化、网络状况变化等因素，VIN 码路由在这方面表现更为灵活。

3. 负载均衡能力

虽然DNS可以实现一定程度的负载均衡，但其能力有限，难以根据实时的系统负载情况进行动态调整。VIN 码路由可以更容易地实现复杂的负载均衡策略，如考虑车型、使用频率等因素。

4. 安全性考虑

DNS 系统容易受到 DNS 劫持、缓存污染等安全威胁。而基于 VIN 码的专有路由系统可以实现更高级的安全措施，更好地保护敏感的车辆数据。

5. 性能和延迟

DNS 查询可能引入额外的网络延迟，特别是在跨网络或跨地域的情况下。VIN 码路由可以更快速地进行路由决策，减少延迟，这对于需要实时响应的车联网应用至关重要。

6. 扩展性限制

随着车联网规模的扩大，DNS 系统可能面临性能瓶颈。VIN 码路由可以更容易地水平扩展，适应不断增长的车辆数量和数据流量。

7. 自定义能力

DNS 系统的路由规则相对固定，难以实现复杂的自定义逻辑。VIN 码路由可以根据业务需求灵活定制路由策略，如基于车辆使用模式、地理位置等因素。

8. 故障恢复能力

在系统故障或网络中断情况下，DNS 可能需要较长时间来更新和传播新的路由信息。VIN 码路由可以实现更快速的故障检测和恢复机制。

9. 合规性和数据主权

在处理跨地域数据时，VIN 码路由更容易实现对数据存储和处理位置的精确控制，有助于满足不同地区的数据合规要求。

4.3 基于 VIN 码路由机制的网关实现

基于 VIN 码路由机制的构建的网关提供了比 DNS 路由更精细的控制和更丰富的功能，适合作为车辆网单元架构的路由层。

网关在单元架构中作为一个中央分发层，接收所有入站请求并基于 VIN 码机制路由到适当的单元（Cell），实现如下功能：

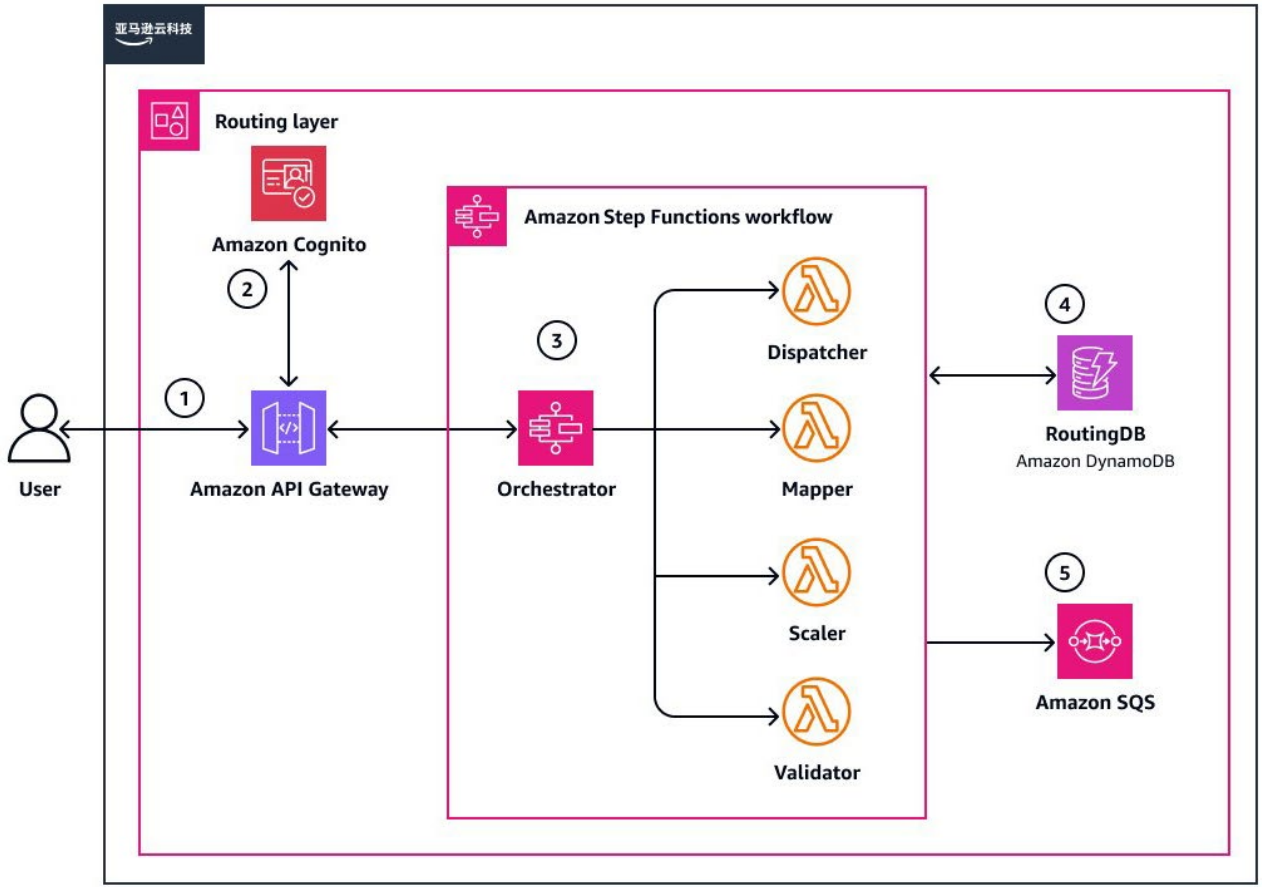
- **接收所有客户端的初次请求**
- **执行请求验证和认证**
- **应用速率限制和流量控制**
- **根据路由规则将请求转发到适当的单元**
- **实现实时健康检查和故障处理**

网关可以采用托管的 Amazon API Gateway 或基于 EC2、EKS、Lambda 等计算服务搭建自建网关。Amazon API Gateway 提供低运维负担、快速部署和可预测的成本结构，有 SLA 保证，天然跨可用区部署，亚马逊云科技负责基础设施可靠性，自动处理底层硬件故障，适合标准 API 需求和资源有限的团队。自建网关提供了灵活性、性能控制和定制能力，适合有足够资源的团队。

存储路由规则和分区映射信息的数据库或配置系统可以由关系数据库或 NO-SQL 数据库（如 Amazon DynamoDB）或 S3 对象存储构成。S3 通常需要读取整个配置， 提供简单、经济且易于管理的解决方案，适合配置相对静态的场景。DynamoDB 可以只读取所需的特定项目，提供更高的性能、灵活性和实时更新能力，适合动态变化的环境。例如使用 DynamoDB 进行活跃映射管理，同时使用 S3 进行配置备份和故障恢复。在车联网场景，推荐采用 DynamoDB 作为存储路由规则和分区映射信息的数据库。

Amazon DynamoDB 是一项无服务器 NoSQL 数据库服务，支持键 - 值和文档数据模型。开发人员可以使用 DynamoDB 构建无服务器应用程序，DynamoDB 可通过自动水平扩缩进行扩展，以支持几乎任何大小的表，并可轻松扩展到互联的 region。

4.4 基于 Serverless 技术的单元架构路由层的参考实现



上图给出了 Serverless 路有层的参考实现，使用 Amazon DynamoDB 存储路由规则，使用 Amazon API Gateway+Lambda 完成路由调度。采用 Serverless 技术只需为实际使用的计算资源付费，能够根据应用的实际需求自动扩展计算资源，部署简单且资源管理自动化，车联网开发者可以更快地测试新特性或更新，并且 Amazon Serverless 服务天然部署在一个区域的多个可用区上，可以应对可用区级故障，基于上述优势，Serverless 技术可以作为路由层的技术实现。

图示中的路由层是单元架构（Cell-Based Architecture）的入口点，负责将车辆分配到适当的单元并提供端点信息，它采用静态路由策略，使客户端在初始登录后缓存端点信息然后直接与单元通信，这种解耦方式确保即使在路由器受损时，单元架构应用仍能保持功能。各个组件负责的功能如下：

API Gateway 作为单元路由层 API 端点的前端，处理所有入站请求并转发到后端服务。	Amazon Cognito 处理用户认证和授权，提供安全访问控制，实际使用中可替换为 Oauth 等实现方法。

DynamoDB 作为 Routing DB，存储单元信息和属性（API 端点、亚马逊云科技区域、状态、指标），维护车辆到单元的映射关系，并使用全局二级索引优化查询效率，表的创建如下（示例）：

VehicleStatus 表

存储车辆状态信息，使用 VIN 作为主键。

RoutingRules 表

存储 VIN 码与业务单元的映射规则。

Amazon Step Functions 工作流作为路由器的核心编排组件，根据资源路径执行不同的 Lambda 函数，协调整个路由过程，具体的 Lambda 函数说明如下：

Dispatcher（调度器）

为新车辆注册并分配静态单元，返回分配的单元的端点信息。

Mapper（映射器）

读取 RoutingRules 表，根据 VIN 码与业务单元的映射规则为已分配车辆提供其端点信息，监控单元 URL 变更并更新车辆设置。

Scaler（扩展器）

跟踪单元占用率和可用容量，当单元达到预定容量阈值时请求新单元，通过 SQS 向配置和部署层发送请求。

Validator（验证器）

验证单元端点并检测潜在问题，提供端点验证状态和必要的修复步骤，增强用户体验，提供相关的错误信息。

Amazon SQS，当需要新单元时，通过队列向配置和部署层发送请求。

上述组件实现了，单元架构路由层的基本功能，具体实现可以参考：

Set up a serverless cell router for a cell-based architecture

参考代码

4.5 路由层的可靠性保障

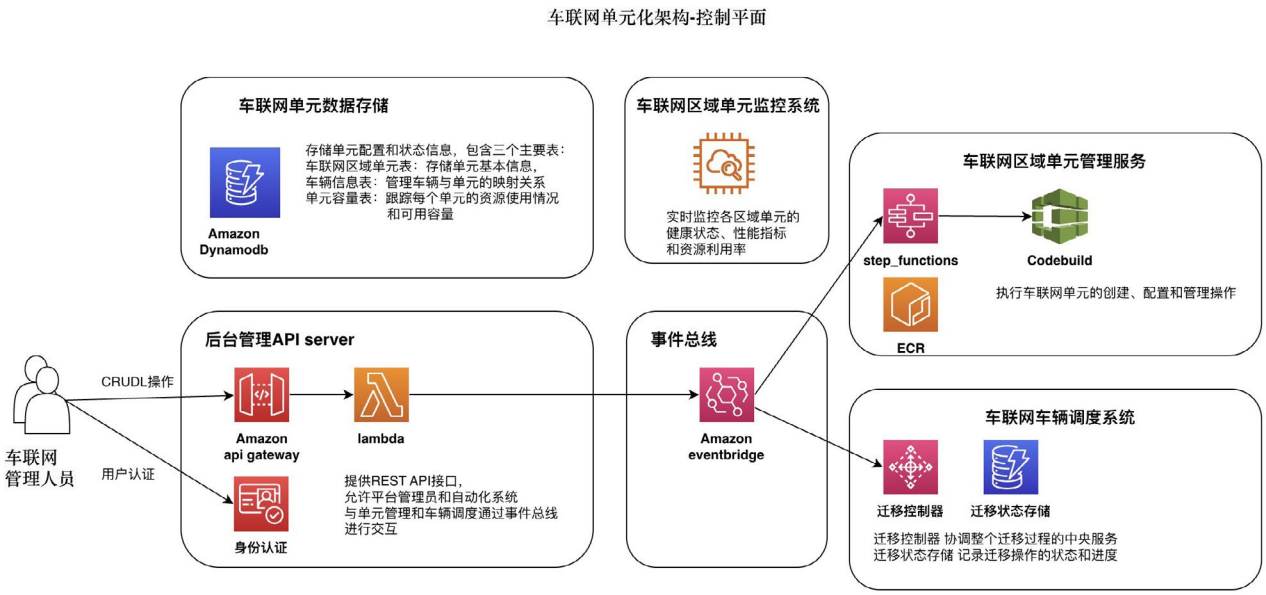
上述路由层可以部署在亚马逊北京区域，由于上述服务都是托管的区域级服务，天然部署在多个可用区上，因此可抵御单个可用区失效的风险。

由于路由层是车联网单元架构的调度中心，因此可以考虑在亚马逊宁夏区域的容灾站点，建议采用**温备的容灾策略**（active-stand-by 模式），即 DynamoDB 中的数据使用 global table 实时复制到宁夏区域，同时在宁夏区域部署 API Gateway 和 lambda 以及 SQS，但没有业务流量，只有在北京区域出现服务中断达到某一阈值时才将流量全部切换到宁夏区域，根据 Serverless 服务的特性，用户在宁夏区域的 API Gateway 和 lambda 以及 SQS 几乎没有使用成本，只支付 DynamoDB 的存储成本和少量用于复制的计算成本。

当车辆数量达到某一数量级后（50 万 +）可以考虑建设跨 region 双活的路由层。此时，由于路由层的应用是无状态的，且数据存储在 DynamoDB 中的 global table 中，故可以使用 Route 53 的地理位置路由策略或延迟路由策略进行路由层的流量分配。

车联网单元化架构的控制平面的实现

车联网单元化架构的控制平面负责实现对各个车联网区域单元的部署、新建、恢复和单元内车辆的迁移（即增添 / 裁撤单元时，车辆迁移到另外一个单元）。控制平面的架构图如下：



车联网单元管理系统由以下核心组件组成：

车联网单元数据存储

推荐使用 Amazon DynamoDB 表存储单元配置和状态信息，包含三个主要表：

- **车联网单元表：**存储单元基本信息，如单元 ID、区域、状态和容量
- **车辆信息表：**管理车辆与单元的映射关系
- **单元容量表：**跟踪每个单元的资源使用情况和可用容量

车联网区域单元管理服务

- 中央控制平面，负责车联网单元的创建、配置和管理
- 提供 REST API 接口，允许平台管理员和自动化系统与单元进行交互
- 实现单元生命周期管理，包括创建、扩展、缩容和删除单元

后台管理 API server

后台管理 API 服务器是接受单元管理系统 API 调用的组件，用于对车联网单元执行 CRUDL 操作以及车辆调度动作。它包含一个用于公开单元管理系统 API 的 API 网关、一组用于运行系统业务逻辑的 Amazon Lambda 函数，以及一个负责对单元管理系统管理用户进行身份验证的身份认证系统。

事件总线

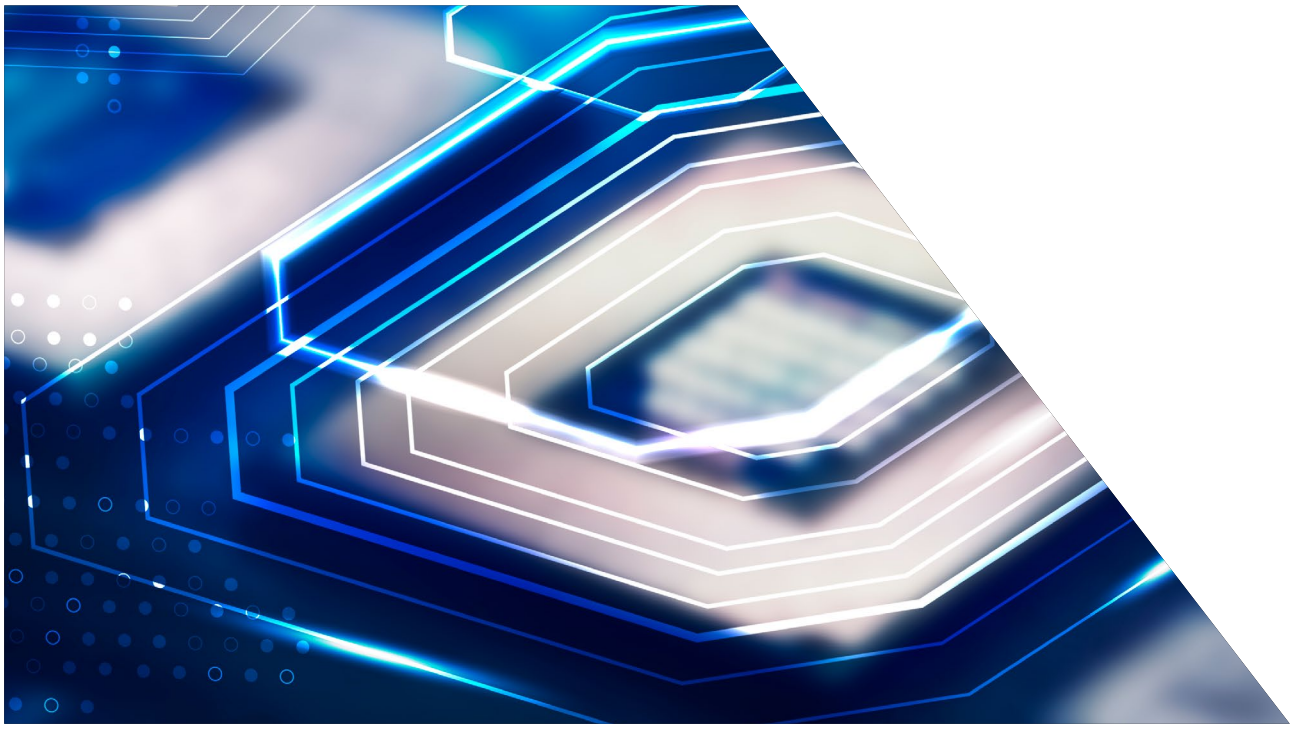
事件总线是一个负责将 API 服务器与单元管理和车辆调度系统解耦的组件，可使用 Amazon EventBridge 实现。Amazon EventBridge 是一种无服务器事件总线服务，可以设置路由规则来确定发送数据的目的地，轻松地将应用程序与来自各种来源的数据相连接。借助 EventBridge，可以构建松散耦合的、事件驱动的分布式架构。

车联网区域单元监控系统

实时监控各区域单元的健康状态、性能指标和资源利用率，并设置自动告警机制，在单元出现异常时触发通知。并收集单元级别的用于容量规划和性能优化，具体包括如下几方面指标，**资源利用率指标、单元运行状态指标、单元负载特征数据、单元容量边界指标、服务质量指标、单元间比较指标，各类指标的参考设置，详见附录。**

车辆跨区域调度系统

一个是增添 / 裁撤单元时单元工作负载的迁移以实现重平衡。具体过程详见 6.3 的说明。

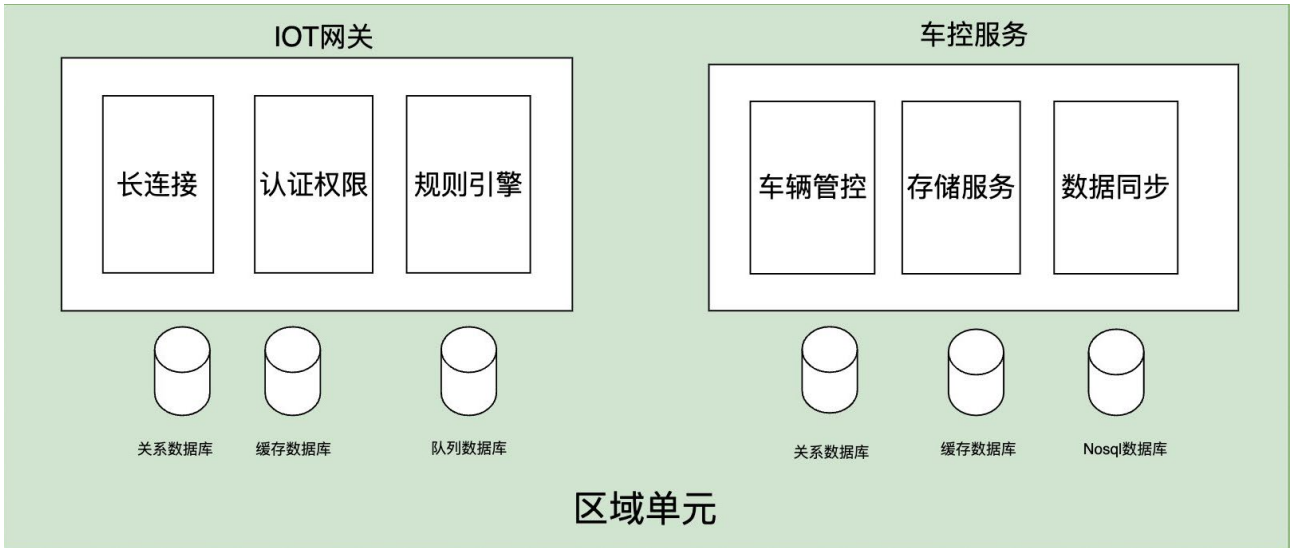


车联网区域单元
业务概览

6.1 车联网区域单元中的应用组成

单元化车联网架构的核心在于其创新的区域单元设计，每个区域单元由两个关键组件构成：**IoT 网关和车控服务**。

这两个组件不仅在构建高效、安全、可靠的车联网系统中扮演着至关重要的角色，更是实现区域单元化设计的关键。由于其业务属性以及所存储数据与车辆特性密切相关，能够有效地与其他业务系统解耦，实现数据在区域内的自闭环，因此可以便捷的基于车辆纬度进行设计。



IoT 网关

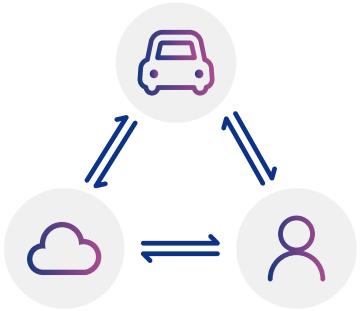
作为车辆与云端服务之间的桥梁，**负责数据的高效传输**、**协议转换**、**安全加密**以及**初步的边缘计算**。它确保了来自车辆的海量数据能够被有效地处理和传输，同时也为车辆提供了与外部世界通信的安全通道。

IoT 网关作为车联网系统的核心组件，集成了**长连接**、**认证权限**和**规则引擎**三个关键功能，共同构建了高效、安全、智能的数据处理和通信平台。

- 长连接机制通过维持车辆与服务器间的持续通信，实现了实时数据传输和即时响应，显著提高了通信效率并降低了网络负载。
- 认证权限模块则负责严格的身份验证和访问控制，通过多重安全机制如 TLS/SSL 加密、token 认证和基于角色的访问控制，确保了系统的安全性。
- 规则引擎作为 IoT 网关的智能中枢，能够灵活处理复杂的业务逻辑，执行数据过滤、事件触发和智能决策等任务，支持从异常检测到预测性维护等多样化的应用场景。

车控服务

提供车 - 云 - 人间的互动交互，其核心目标是提升用户车辆使用的便利性，以及更优的车辆使用体验。如用户通过 App 发送远程控制指令（解锁车门、启动空调、查看车辆位置等），并接收实时反馈（车辆状态、环境信息等）。

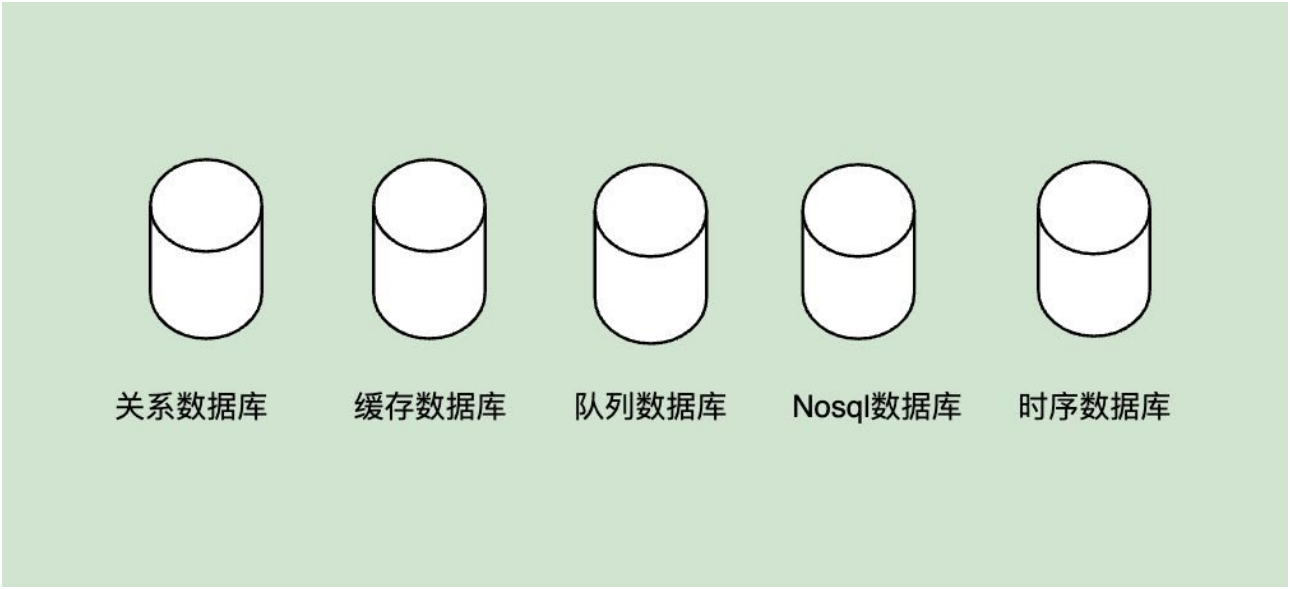


车控服务集成了多项业务服务系统，其中车辆管控服务、支持对象数据上传的存储服务和数据同步服务构成了其核心功能架构。车辆管控服务通过移动应用程序为用户提供了实时监控和操控车辆的能力，涵盖从启动引擎到调节车内设置等多种功能，大大提升了用户体验和车辆管理效率。存储服务采用对象存储 (S3) 结合 STS（Security Token Service）的方式，确保了数据传输的安全性和效率。车辆首先获取临时的 STS 凭证，然后直接将数据安全上传到指定的对象存储桶，这种方法不仅提高了数据传输的安全性，还实现了精细的权限控制。与此同时，数据同步服务负责将分散在各个车辆上的海量数据高效、可靠地传输到中心单元，为后续的大数据分析等其它业务系统提供数据汇总。

*** 注意：**在实际应用中，各家车企会根据自身的业务需求和技术路线，对具体的实现方式和产品选择做出不同的决策。

6.2 车联网区域单元中的常用数据结构

在车联网系统中，不同类型的数据库扮演着关键角色，每种数据库都有其特定的用途和优势。多种类型的数据库协同工作，使得车联网业务系统能够高效处理从实时车辆控制到复杂数据分析等各种场景，为车辆基本信息、实时状态、遥测数据、性能指标等多样化数据提供合理的存储和处理。



关系型数据库：如 Amazon RDS MySQL 主要用于存储结构化的车辆和用户信息，管理系统配置和处理复杂的业务逻辑。

缓存数据库：如 Amazon ElastiCache（或 Redis）则通过存储临时会话数据和热点信息，显著提升了系统的响应速度。

队列数据库：如 Amazon MSK（或 Kafka）在处理高并发数据流和管理异步任务方面发挥着关键作用。

NoSQL 数据库：如 Amazon DynamoDB （或 MongoDB）凭借其灵活的数据模型，成为存储和处理大量非结构化数据的理想选择，特别适合处理车辆遥测数据和用户行为日志。

时序数据库：如 Amazon Timestream（或 InfluxDB）则专门用于高效存储和分析时间序列数据，对于跟踪车辆性能指标和环境传感器数据至关重要。

在车联网系统的区域单元架构中，数据服务的可靠性是保障整体系统稳定运行的关键基石。由于车联网系统需要实时处理和响应大量车辆数据，包括位置信息、行驶状态、故障诊断等关键数据，因此数据服务的中断或异常将直接影响车辆的正常运行和用户体验。为此，区域单元中以上的各类数据服务都必须采用高可用的架构设计，为车联网系统提供稳定可靠的数据支撑。亚马逊云科技的各类数据服务通过多可用区部署、数据多副本同步、自动故障转移和自动扩展等机制，实现了高可用性和数据持久性，满足不同类型数据库和缓存的业务需求。

以下是亚马逊云科技中 Amazon RDS、Amazon ElastiCache、Amazon MSK、Amazon DynamoDB 及 Amazon Timestream 等服务实现数据高可用性的主要机制和特点：

Amazon RDS（关系型数据库服务）

- 通过 Multi-AZ（多可用区）部署实现高可用性。在不同可用区部署同步备用实例，主实例故障时自动故障转移，通常 60 秒内完成且无数据丢失。
- 支持多个读副本扩展读取能力，同时 Multi-AZ DB Cluster 结合了高可用性和读取扩展功能。
- 使用同步复制技术保证主备实例数据一致性，备份不会影响主实例性能。
- Aurora 版本支持跨区域的 Global Database，实现跨区域高可用和低延迟读取。

Amazon ElastiCache（内存缓存服务）

- 提供 Memcached 和 Redis 两种缓存引擎，Redis 支持复制、持久化和事务，适合需要丰富功能和高可用性的场景。
- 通过主从复制和自动故障转移机制实现高可用，支持多可用区部署，保证缓存服务的可靠性和容错。

Amazon MSK（托管 Apache Kafka 服务）

- 集群节点均衡分布在多个可用区，避免单点故障和可用区级故障，默认跨 3 个可用区部署。
- Kafka 数据通过多副本机制（如默认副本因子为 3）和多数确认机制保证消息持久性和可靠性。
- 自动检测和快速替换故障节点，支持自动补丁和升级，确保集群持续健康运行。

Amazon DynamoDB（NoSQL 数据库）

- 数据自动跨多个可用区复制，提供 99.999% 的高可用性和耐故障性。
- 支持自动扩展容量，保证在高负载下依然稳定快速响应。
- 作为完全托管的服务，无需用户管理服务器和补丁，简化高可用架构的实现。

Amazon Timestream（时序数据库）

- 设计用于高性能、可扩展的时序数据存储，支持将近期数据保留在内存中以加速访问。
- 通过亚马逊云科技底层多可用区架构保证数据的高可用性和持久性。

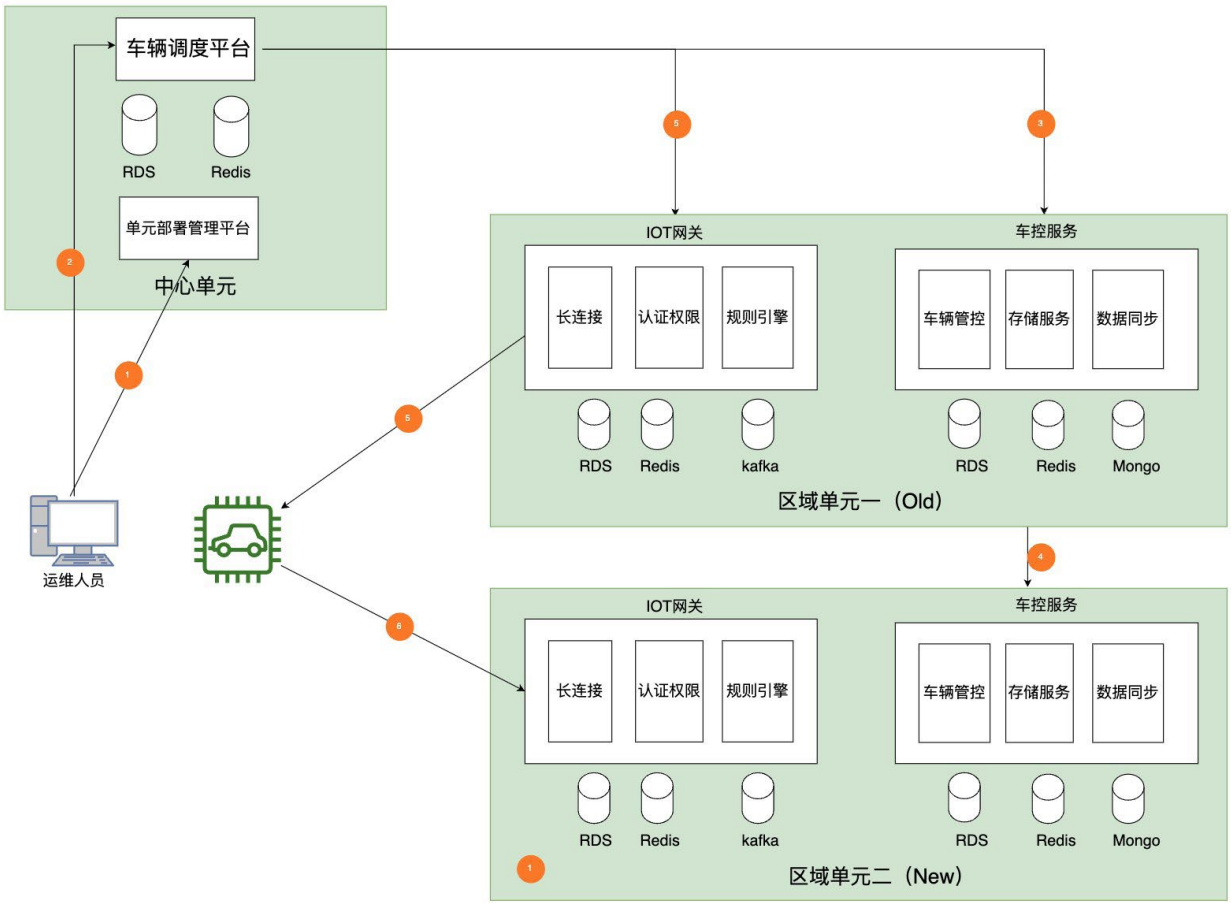
6.3 单元的工作负载的迁移和容灾

单元架构内的车辆工作负载在两种场景下，需要进行调整，一个是增添 / 裁撤单元时单元工作负载的迁移以实现重平衡（rebalance）的目的，一个是单元的容灾的场景。

增添 / 裁撤单元：

随着车辆网系统的容量扩展和升级，可能会出现增添新单元以及裁撤旧单元的需求，此时工作负载和单元之间的映射关系会发生改变，存在工作负载在单元间迁移的过程，这一过程最好是对外透明的，只影响正在迁移的工作负载。这个功能需要路由层配合控制平面的迁移控制器实现，在工作负载迁移期间，路由层把请求转发到对应的单元。

车辆单元调度流程： 车辆从单元一迁移到单元二



1. 运维人员通过单元部署管理平台在新的单元中创建所需要的区域单元服务，以及数据库资源
2. 运维人员通过车辆调度平台，更新要切换车辆所属单元的信息，同时车控服务进行”车辆锁定”流程设置，此状态下车辆将暂时停止数据上传，车控功能有分钟级的不可用，对用户侧无感。建议根据自身业务系统的负载情况，在业务低峰期进行操作
3. 运维人员通过车辆调度同步迁移车辆数据库中的实时数据到新区域单元的数据库中，保障车辆单元切换后已有的数据为最新状态
4. 运维人员通过车辆调度平台更新车辆的长连接连接状态信息 (topic), 主动触发车辆重新连接流程
5. 更新车辆的长连接状态信息 (topic)，主动触发车辆重新连接
6. 被切换的车辆通过车辆连接流程，连接到新的单元，至此车辆完成所属单元的切换，详见车辆连接流程

容灾：

当车辆网的某个单元，需要对单元范围的故障需要有应对的方案，根据不同的成本，采用单元的就地恢复和单元的容灾恢复两种方法，对于某些为核心用户服务的单元建议采用容灾恢复的方法，即为某单元构建灾备单元，工作负载常态下在主单元中，旦主单元发生不可用故障，需要能够快速切换到灾备单元中，由灾备单元在设定的 RTO 时间内接管车辆网业务。

亚马逊云科技提供从流量调度、应用多区域部署、数据复制、数据库复制等多个层面的容灾服务，可以高效的实现容灾系统的建设，详见，此处不再赘述。

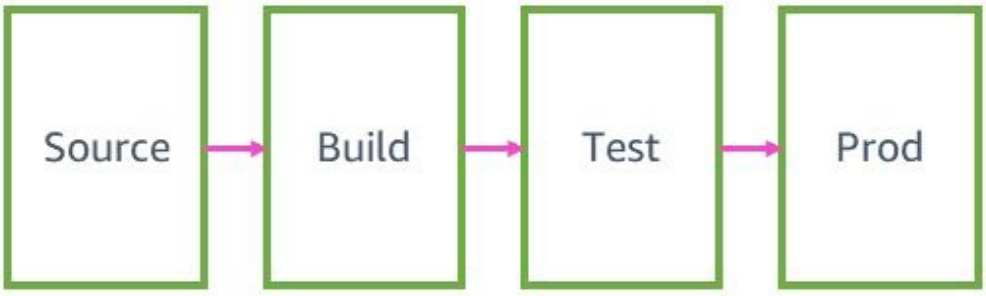
车联网单元化架构的部署策略

07

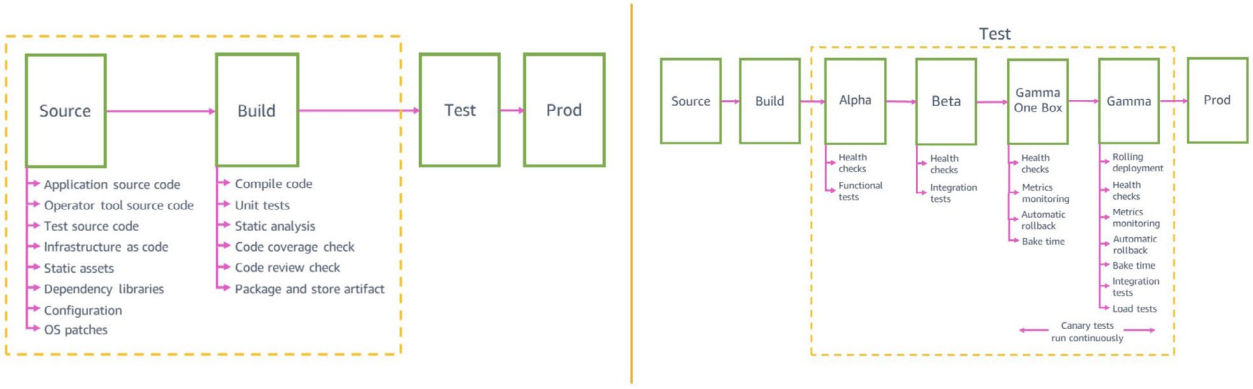
软件的大规模部署是车联网单元化系统中必须面对的问题，根据工信部混沌工程调研报告，错误的配置和软件 bug 是导致企业业务中断的前两大因素，对于车联网系统的软件部署策略需要非常慎重，需要平衡安全性和速度的同时考虑车联网不同应用的特点持续地将软件部署到生产。

简单来说将软件部署分为，涉及车辆驾驶的**安全关键型系统**（如 ADAS、VCU 单元），涉及信息娱乐与连接系统 (IVI、TCU)，以及云端服务系统软件部分，本章将阐述云端系统软件部分。

对于云端服务系统软件的部署，建议设置持续部署管道来快速安全地进行部署。
典型的持续交付管道包括四个主要阶段 – **源**、**构建**、**测试**和**生产**。



Source、Build 和 Test 阶段的示意图如下：



在预生产环境中测试部署及集成测试

在部署到生产环境之前，建议在多个预生产环境（例如，Alpha、Beta 和 Gamma）中部署并验证更改。Alpha 和 Beta 通过运行功能 API 测试和端到端集成测试来验证最新代码是否按预期发挥功能。Gamma 验证代码既能正常发挥功能，又可以安全地部署到生产环境。Gamma 与生产环境相似，包括与生产环境相同的部署配置、相同的监控和警报，以及相同的持续 Canary 测试。Gamma 建议在多个单元内部署，以捕获由于区域差异而造成的潜在影响。集成测试的目的是，在部署到生产环境之前捕获服务的所有意外或错误行为。集成测试既运行正面测试案例，也运行负面测试案例。

生产部署

对车联网单元架构系统进行生产部署的首要目标是防止同时对多个单元造成负面影响。限定每个单独部署的范围将会限定部署失败对客户的影响范围。为了限定部署的范围，建议将管道的生产阶段分为多个阶段，但随着车联网规模的扩大，也需要考虑部署的效率。故建议采用效率和安全兼顾的 Wave 部署策略，即按单元或单元组进行部署，而非同时部署所有单元。从少量单元开始，逐步扩大部署范围，在每个阶段之间设置观察期，监控性能和错误。通过这种 Wave 部署模式，车企可以最小化部署风险，同时保持快速迭代的能力。单元的部署顺序可以按照单元内的客户数量进行排序，客户数量少的发布顺序在前，或根据其他规则对单元的重要性进行排序，重要性高的在后。

如果要进一步提高部署的安全，每个生产 wave 的部署可以从 one-box 阶段开始。即每个生产 one-box 阶段都将最新代码部署到每个 wave 的单元中的一个 box（最多不超过 10% 的用户）中。如果更改在一个 box 中造成了负面影响，则管道会回滚更改，并且不会将其扩展到产品的其余阶段。

这种部署方法的将单元架构的故障隔离原则扩展到了部署过程中，确保部署问题的影响范围有限，从而提高整体系统的可靠性。

在物理部署形式上，以部署在亚马逊宁夏区域的车联网单元为例，为实现上述部署的高可用，推荐应用系统、数据库等系统部署在亚马逊的三个可用区上，任何一个可用区的故障不会对车联网单元承载的业务产生影响。

总结与建议

08

单元架构为新能源汽车车联网系统带来了多方面的核心价值。它显著提高了系统的响应速度和可靠性，通过将车辆连接到地理位置相近的单元，减少网络延迟，同时分布式设计增强了系统的整体稳定性。这种架构优化了资源利用，支持根据区域需求灵活调整资源分配，提高系统效率。单元化设计还为大规模扩展提供了便利，能够通过增加单元数量来满足不断增长的车辆接入需求。

在数据处理方面，单元架构将处理下沉到靠近数据源的位置，实现了更快速的分析和决策，为车辆提供更智能的服务。

在数据处理方面，单元架构将处理下沉到靠近数据源的位置，实现了更快速的分析和决策，为车辆提供更智能的服务。

对于新能源汽车特有的需求，单元架构在能源管理方面表现出色，能够更精确地监控和管理区域内充电设施的使用情况，优化充电策略，提高能源利用效率。系统的可维护性也得到了提升，单元化设计使局部维护和升级变得更加容易，减少了对整体系统的影响。

总的来说，单元架构通过其分布式、可扩展、高效的特性，为新能源汽车车联网系统提供了强大的技术支撑，有助于构建更智能、更可靠、更高效的车联网生态系统。它不仅解决了当前车联网面临的诸多挑战，还为未来的发展和创新铺平了道路，是推动新能源汽车和智能交通发展的关键技术基础。

附录

单元架构和容灾（双活）的区别：

单元架构经常和容灾中的双活策略进行对比，**单元化架构**和**双活架构**都是为了提升系统的高可用性和可靠性而设计的分布式架构，但它们的目标、应用场景和设计理念有所不同。（双活架构一般分同城双活和异地双活，距离越远，难度越大，一般对应云平台的 region 间双活，和多 AZ 高可用部署或多 AZ 容灾部署（很少出现））。

单元化架构和容灾双活架构的区别如下：

容灾场景下多活的定义，可以参考中国人民银行发布的《金融信息系统多活技术规范金融信息系统多活技术规范》中对多活的描述，据此给出 单元架构和容灾（双活）的区别，两者可以结合使用，单元架构可以部署在异地双活的数据中心内部，以实现更加细粒度的业务隔离和故障容忍，通常推荐每个不同的单元都具备独立存活的能力。

	单元架构	双活架构
目标和应用场景	主要目的是提高系统的**可用性**和**扩展性**，通过将系统划分为多个独立单元来实现故障隔离和业务分离。单元化架构更关注的是如何应对**业务复杂性**和**用户规模的增长**。 典型应用场景包括大规模互联网平台、电商平台、金融系统等。这些系统通常需要应对**高并发**和**复杂的业务逻辑**。	主要目的是通过在**不同地理位置**部署两套完整的系统，使得两个数据中心同时对外提供服务，实现地理隔离的同时，提供极高的容灾能力。即便一个数据中心完全失效，另一个数据中心可以接管所有流量。 典型应用场景包括全球性的互联网公司、跨国金融机构等，需要在**大规模的灾难**（如地震、火灾、云服务的区域级故障等）下保持业务连续性。
架构设计	将系统按**业务**或**用户群体**划分为多个独立的单元。单元之间相对独立，故障或流量波动只会影响特定单元，其他单元不受影响。 单元内部可能会使用相同的基础设施，路由层负责将请求按规则分发到不同的单元。	在不同地理位置部署两套**完全独立且同步**的数据中心。这两个数据中心对外提供同样的服务，并且互为备份。 异地双活通常依赖于**数据同步**和**负载均衡**技术。 数据中心之间的数据同步、事务一致性管理是双活架构的重点和难点。

	单元架构	双活架构
故障处理	当一个单元故障时，其他单元依然可以正常运行，避免系统整体不可用。需要在本单元内就地恢复，或通过负载均衡或路由机制将流量引导到正常运行的单元。	当一个数据中心出现故障时，另一个数据中心立即接管所有的流量，实现**无缝切换**，对外表现为**业务持续可用**。
数据一致性	单元之间通常使用**独立的数据存储**，跨单元的数据一致性是一个挑战，尤其在全局事务和全局数据同步方面，需要额外的协调机制。或将数据传输到统一的中心点处理。	由于数据中心在不同的地理位置，数据同步是架构的核心问题。通常会依赖数据库复制、分布式事务等技术来确保数据在不同数据中心间的一致性。业务数据在**多个子系统中存在数据副本**。
扩展性	可以通过增加单元的数量轻松实现水平扩展，非常适合用户规模不断扩大的系统。	主要是通过多个数据中心共同对外提供服务来承载流量，数据中心的扩展相对复杂，通常需要考虑地理位置、网络带宽等因素。
就近服务客户	二者都具备就近服务客户的能力，单元架构通过部署独立的单元实现。	二者都具备就近服务客户的能力，双活架构需要考虑多站点统一考虑。

参考文档

车辆网白皮书 2023（中国信通院）

https://www.caict.ac.cn/kxyj/qwfb/bps/202312/P020240326618179274556.pdf

北京地方标准《车路云一体化信息交互技术要求第 2 部分：应用平台与云控平台》

https://dbba.sacinfo.org.cn/attachment/downloadStdFile?pk=196bd25b6322dcf3691186fe4ac2e60696e5cbc37a48f3eff21c8f9363145ef7

Guidance for Cell-Based Architecture on Amazon

https://aws.amazon.com/solutions/guidance/cell-based-architecture-on-aws/?did=sl_card&trk=sl_card

Guidance for a Cell-Based Architecture for Amazon EKS

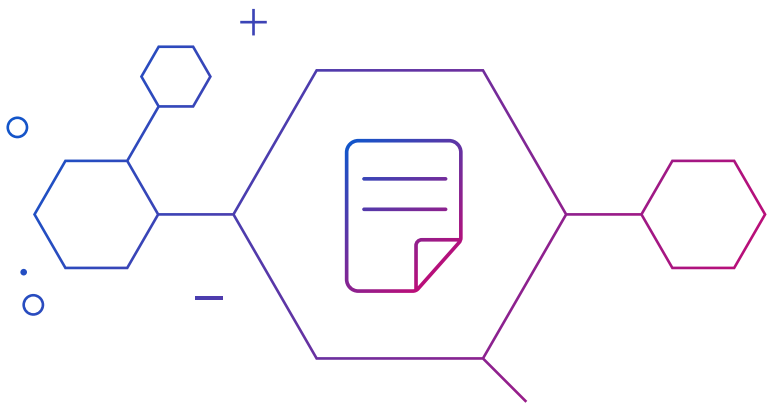
https://aws.amazon.com/solutions/guidance/cell-based-architecture-for-amazon-eks/

Amazon 如何构建和运营软件

https://aws.amazon.com/cn/builders-library/automating-safe-hands-off-deployments/

金融信息系统多活技术规范金融信息系统多活技术规范

https://static1.tianyancha.com/czd_file/standard/28e10edeb122c74a7c3a10459172ca68.pdf



亚马逊云科技