21天微服务实战营-Day1

华为云DevCloud & 微服务联合出品

Day1 微服务架构知识介绍

- 微服务简介
- 容器与容器平台
- 微服务架构模式
- 微服务开发框架
- Service Mesh
- 微服务平台

什么是微服务

微服务架构是一种架构模式，它要求开发者以一种不同于以往的开发方式进行软件开发，设计功能比较单一，拥有接口的服务，他们都可以被独立的构建，测试，部署。

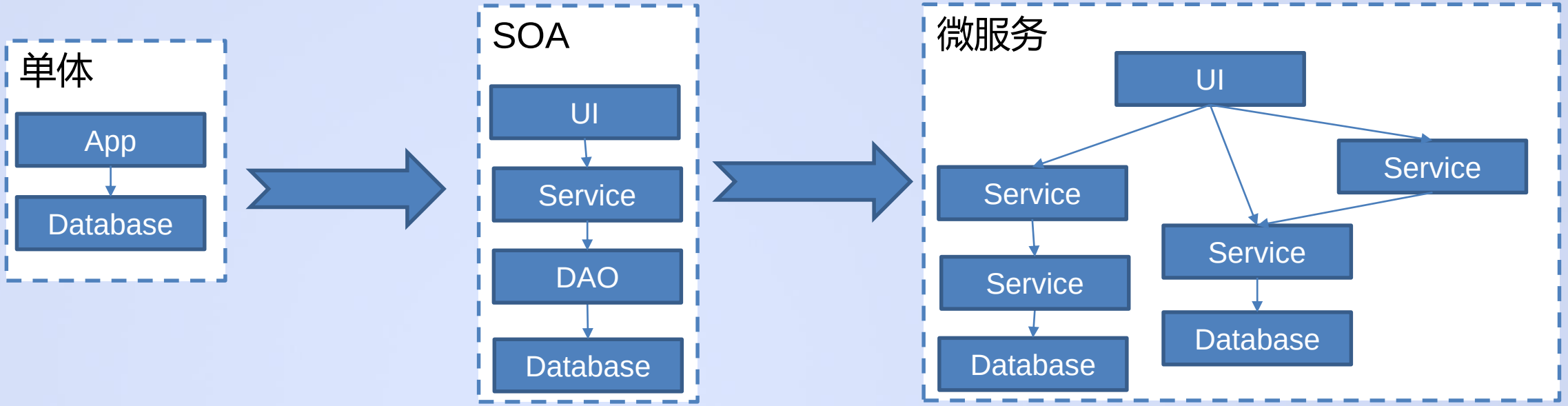
微服务是得益于DevOps文化的发展，持续集成工具的成熟，越来越多的公司向敏捷转型，微服务架构模式可以指导企业开发出具有可伸缩，弹性，高可用的系统，从以往的几个月的上线频率，缩短为几周甚至几天。

传统软件是由单一服务构成，微服务提倡将一个软件按照功能模块进行划分

为什么使用微服务

- 独立运行：服务异常不再彼此影响，必要时将非核心功能隔离，不影响主要功能运转。一个服务实例崩溃不会影响其他实例，整体系统依然正常。按功能伸缩，当某个模块算力需求变化时只进行该功能实例的伸缩，而不是整个系统的伸缩，减少资源浪费。
- 独立升级：一个小特性的更改或者bug fix不会影响大部分功能的正常运转
- 代码复用：一套代码可以用于不同的独立系统中，在公司内部或者开源社区中进行分享。比如，支付服务，用户管理服务，认证鉴权。
- 技术演进：单体服务使用陈旧的技术，想象你过去使用struts1+spring，你想升级struts2来获得一定的收益，接着你想整体切换到Spring MVC,彻底摆脱struts框架，不断地切换框架为工程稳定性带来风险，而陈旧的框架又无人维护。而微服务项目不受旧代码拘束。
- 语言限制：当你发现某个新功能更适合使用Go而不是java时该怎么办，Java也不是万金油，每种语言都有适合自己的场景，微服务使开发者能根据服务场景选择语言。招聘开发者也不必局限于语言
- 团队：小团队运作更加敏捷，配合紧密，开发周期短，组织扩张灵活

历史



微服务的演进历史是漫长的，从单体的MVC架构到分布式SOA架构，在结合了敏捷开发，DevOps等理念后最终诞生了微服务。一个很好的印证是，在我深入的实践了DevOps和敏捷开发后，自发地开始萌芽了微服务的思想理论。

微服务最早出现在国内是在2015年初的时候，成功的案例有AWS以及Netflix等公司。

EC2最早是亚马逊内部使用的一个服务，最终被作为一种服务对外提供成为AWS，而基于微服务架构，AWS基于现有的服务之上快速迭代新的产品，丰富AWS能力，现在已经拥有100多种不同的服务，回报是巨大的。

华为很早便践行了微服务理论并对外开源了微服务相关项目，华为云得益于微服务架构快速推出大量新的云服务。

微服务面临的挑战

1. 持续集成：大量的工程，需要一个持续集成工具自动完成编译，打包，发布，部署等工作
2. 版本管理：大量的版本，就会遇到兼容性问题。你需要让项目可控
3. 文档管理：版本在持续升级，服务接口不匹配。你需要一个文档管理系统，并让开发者严格遵守文档进行开发
4. 生命周期管理：服务运行期，需要一个平台管理服务，除了部署，启停，还要能够在服务崩溃时自动拉起服务
5. 运维：运维人员操作服务，查看指标，日志，分布式调用链，更改配置项都由于微服务架构而变得比以往更加复杂
6. 调试：在开发期你如果依赖于很多微服务，如何方便地在本地去调用依赖的服务。
7. 网络调用：从过去本地的内存栈调用变为了网络调用，不再可靠
8. 安全：如何控制不让未经授权的调用者访问到自己的数据
9. 如何云服务化：转型微服务涉及一系列的工作，处理以上复杂的问题需要大量的基础代码研发，如何能驾驭诸多的技术和文化变更

构建微服务系统是困难的

接下来的章节中将简单介绍微服务模式带来的问题的解决方式，并在后续课程中进行实战

容器与容器平台

可以学习day1-day4的内容来了解相关技术

https://activity.huaweicloud.com/21days_cce/index.html

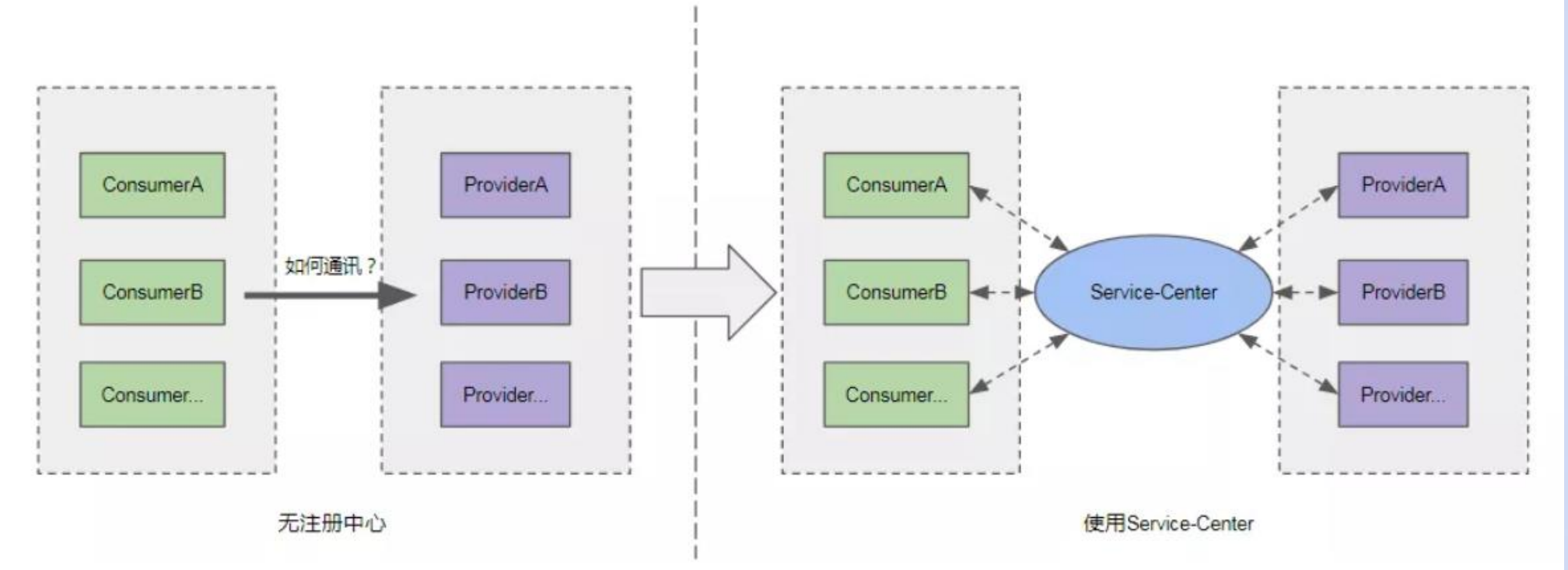
容器的出现帮助了微服务技术体系的成熟，这是至关重要的一环

容器和微服务是天生一对，在过去的虚拟机时代中，VM的启停甚至长达数分钟，在进行弹性伸缩时，假设你设置了这样的伸缩条件“CPU负载高于80%时，新增一台机器 “，那么很可能在你没有得到一台新机器前，现存的VM中的服务就已经被压垮，继而引发级联崩溃。

容器的启停只需要数秒，并且由统一的容器平台管理全生命周期，为微服务系统提供了运行条件

从以往以虚拟机为中心的管理变为了以服务为中心的管理。让开发者专注于应用本身

微服务模式—注册发现



注册发现是微服务的基础，每个微服务实例都要注册自己的信息与地址到注册中心中，每个实例都在注册中心中查询自己需要访问的实例信息与真实地址。

Service center是华为云提供的一个典型的注册中心，帮助服务间进行注册与发现

注册中心优点：

1.解耦服务提供者与服务消费者，服务消费者不需要硬编码服务提供者地址。

2.服务动态发现及可伸缩能力，服务提供者实例的动态增减能通过注册中心动态推送到服务消费者端。

3.通过注册中心可以动态地监控服务运行状态

微服务模式—路由管理

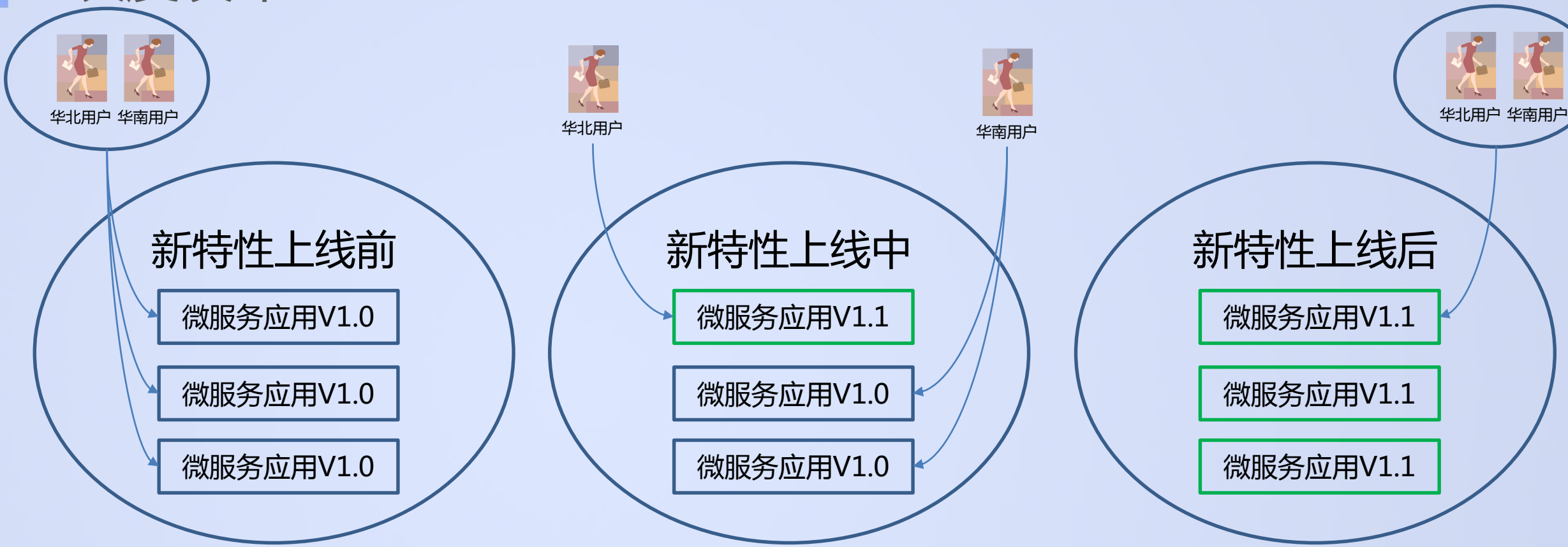
ServiceA

微服务	地址	版本	数据中心
ServiceB	10.4.1.1:8080	1.0	北京
ServiceB	10.4.1.2:8080	1.0	深圳
ServiceB	10.4.1.3:8080	2.0	北京

微服务A已经根据微服务名查询到了相关的实例信息并放置于本地内存，如上图表格，那么如何决定自己到底要选择哪一批实例进行访问呢？

可以看到每个实例都具有一定的属性，上表中有2个属性版本与数据中心。那么编写算法根据微服务名和属性筛选服务实例，最终就可以决定要访问的服务实例的集合了

灰度发布



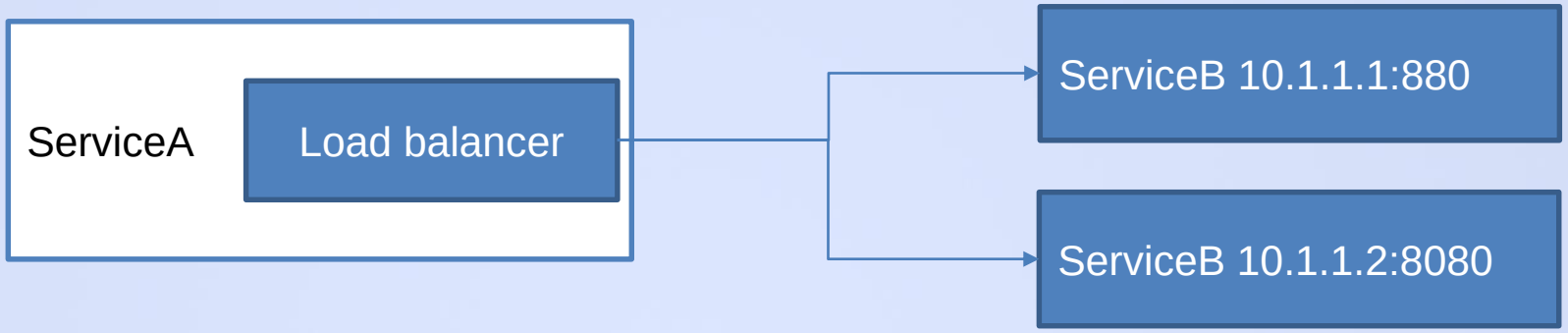
路由管理可以为我们解决什么问题？

灰度发布就是一个典型的场景

原本系统中运行着1.0版本的服务，我们可以将部分用户的流量迁移到新版本1.1中

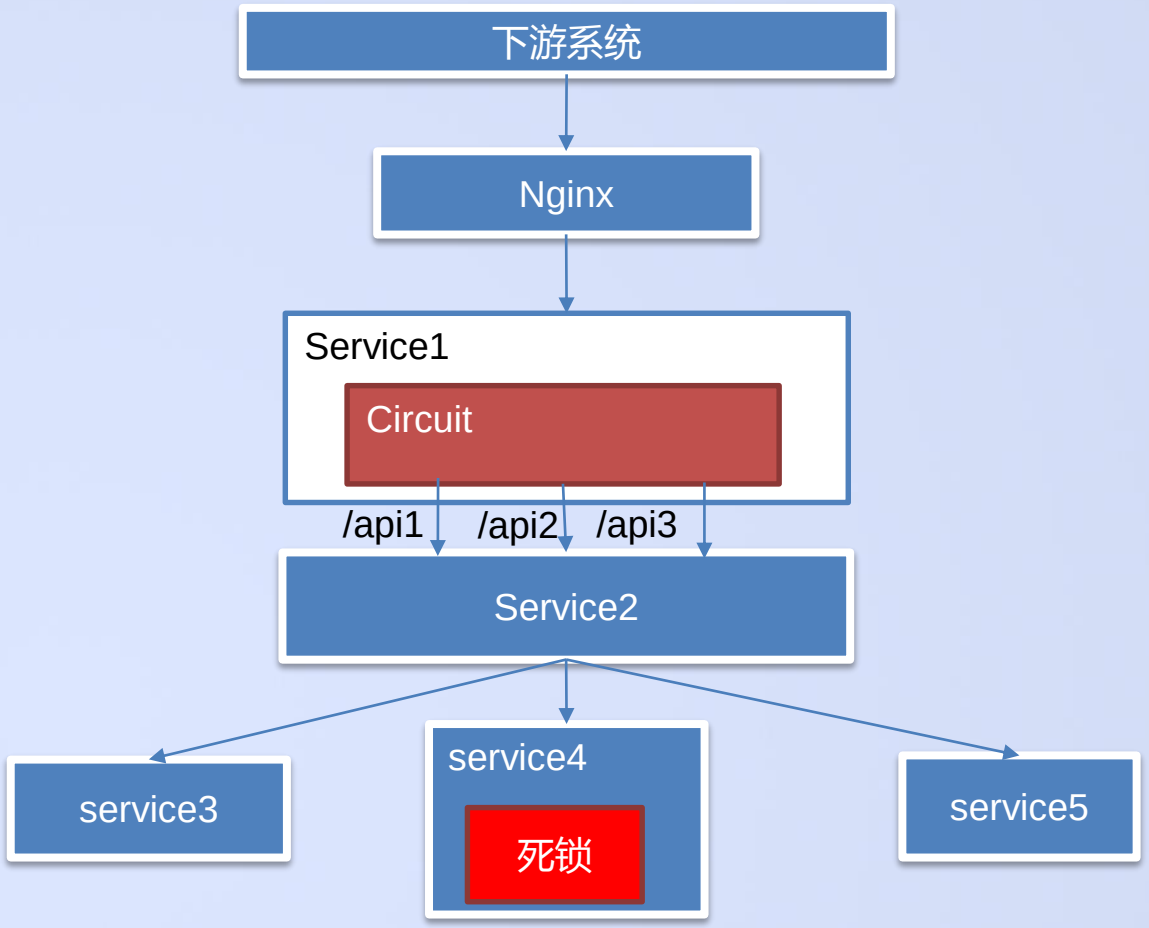
让部分用户优先试用。最终将所有流量迁移到新版本中

微服务模式—客户端负载均衡



当在路由过程中决定了一个实例集合后，就可以对集合实行负载均衡算法，选择其中一个实例访问，即客户端负载均衡，区别于nginx这样的传统负载均衡组件，负载均衡直接在客户端，也就是你的业务进程内部进行。

微服务模式—熔断



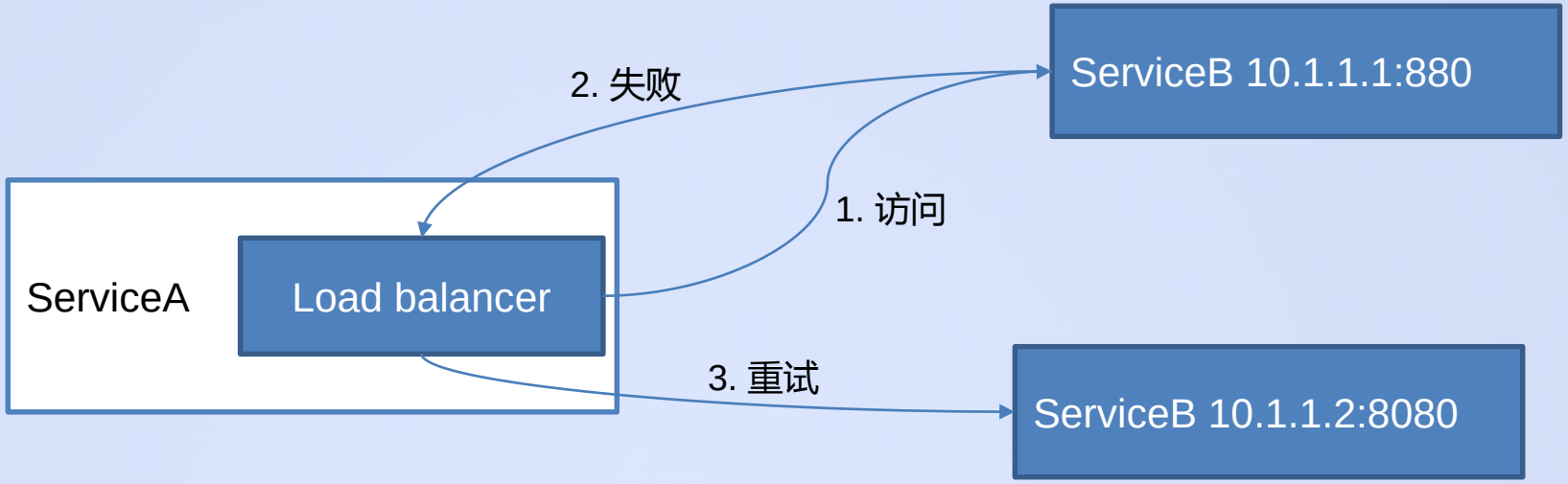
当服务发生错误，超时等问题，系统需要将这部分非核心功能隔离，以免引起级联崩溃

上图，Service1，调用api2，访问service4完成一个业务，当这个api2出现死锁时，将引起下游未做超时处理服务沾满线程池，最终大范围瘫痪，导致其他功能失效，想想如果更加庞大的系统，是一个多大的灾难，异常将被放大。

如果能够在访问api2达到一定超时次数就将api2隔离掉，不再发生网络调用，那么就不再产生新的死锁，系统稳定性就会提升

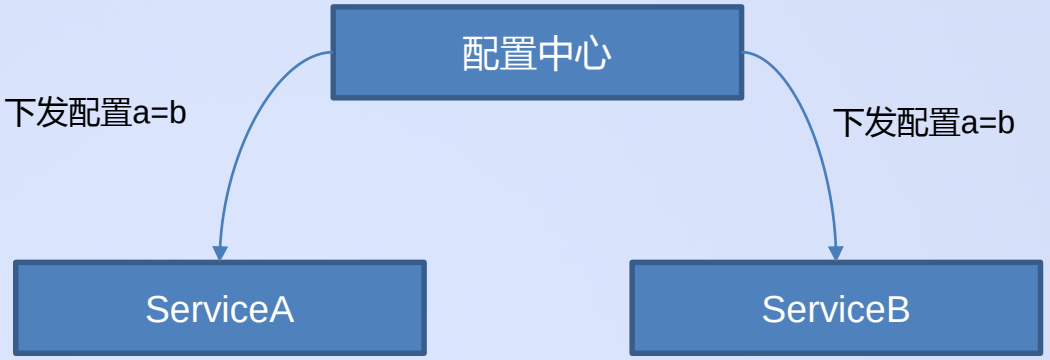
道理就像战舰的隔离仓，当遇到漏水的船舱时要将有问题的船舱隔离以避免灾难蔓延。

微服务模式—容错



当一个请求失败时，可以尝试对同一个地址进行重试或者从负载均衡中选取一个新的地址进行重试。

配置管理



服务分布在各个服务器中，在需要更改配置时，我们不想登录到每个机器上进行文件编辑，并进行服务重启，配置中心能够解决这个问题

当你遇到配置变更时，只需要在配置中心中进行更改，各个服务受到变更消息后，进行更改并在运行时生效

Day5将详细介绍

监控

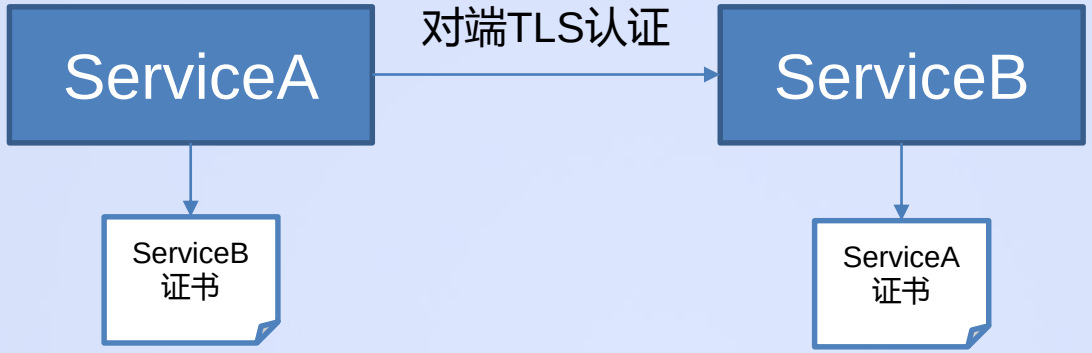
日志：分布式系统庞大，无法再传统的登录到服务器上去查看日志，需要将日志上报并统一汇聚到一个监控系统中

分布式调用链追踪：网络的调用不同于本地调用，就像本地调用可以用工具分析，分布式的网络调用也需要被监控起来，并在监控系统中进行分析以便随时掌握系统调用状况，分析服务性能瓶颈等。

指标：指标分为通用指标如cpu，内存，请求数量，延迟等。以及自定义指标如用户注册量，商品购买量等

华为的APM服务是一个监控平台，能够支撑微服务系统的运行。开源中有Promethues，Zipkin，elastic search，kafka，TSDB等大量服务可用于搭建一个监控系统。

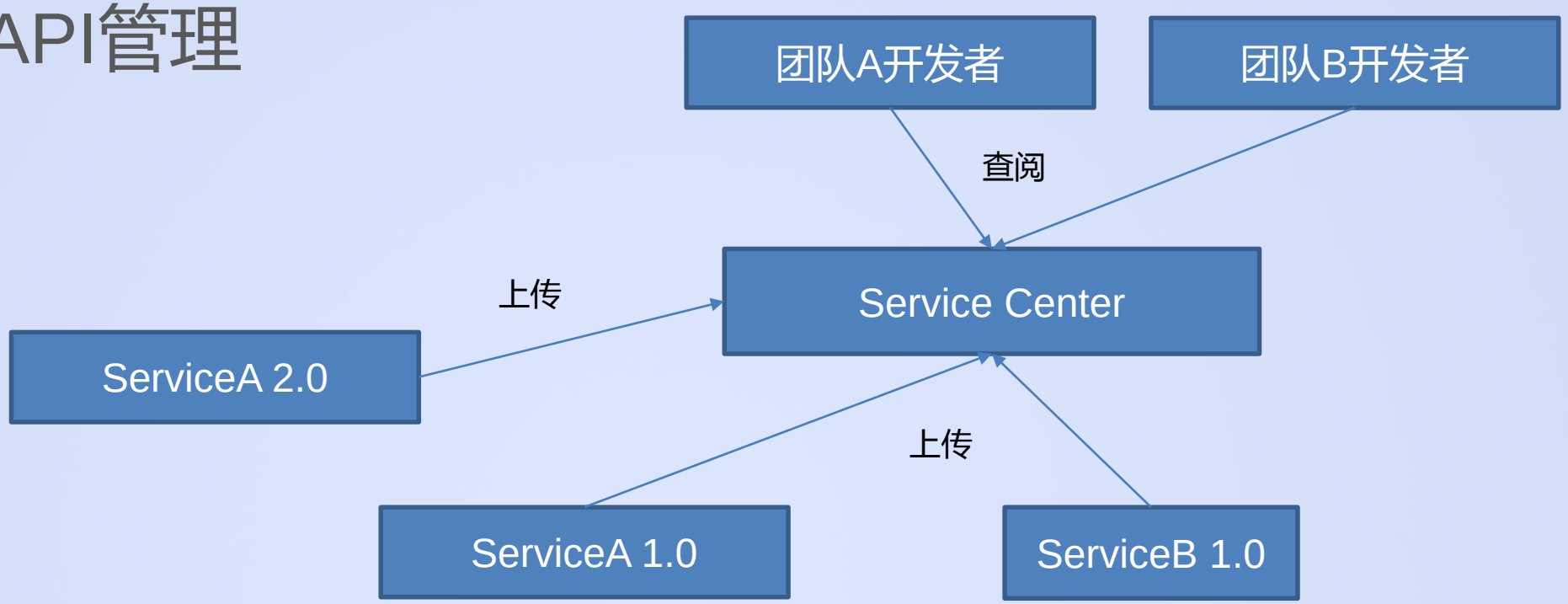
安全



网络调用引起了安全访问的问题，使用对端TLS认证可以解决这个问题

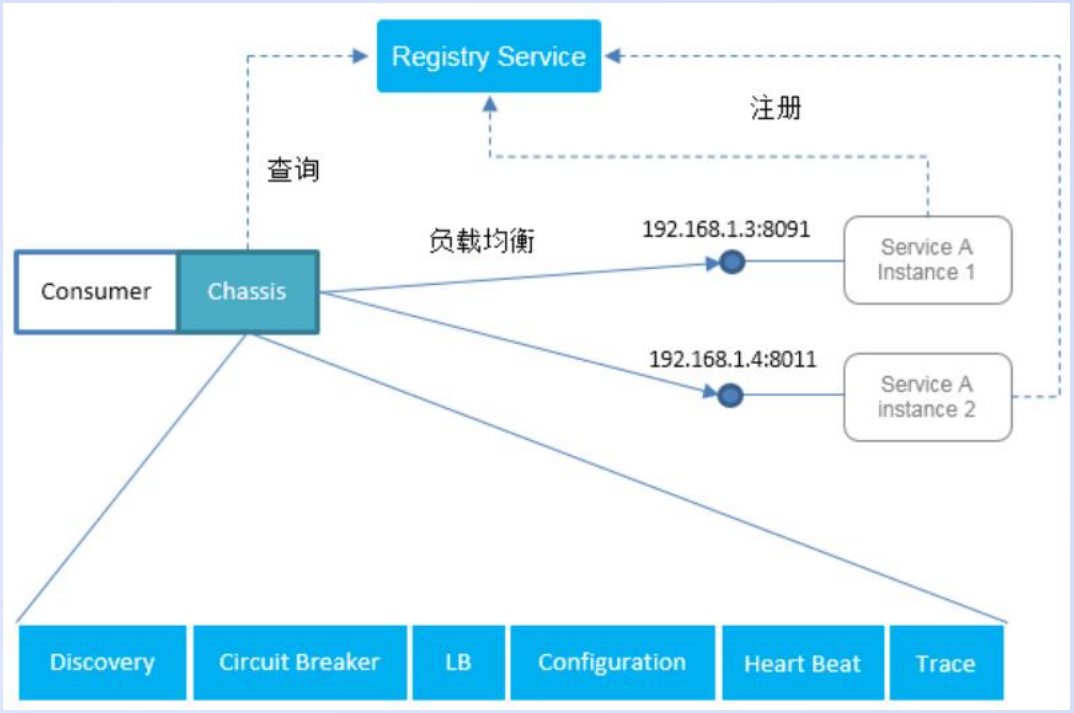
由ServiceB，ServiceA的开发者都签发证书，分发给彼此，两者加载对方证书，对彼此进行认证，以确定彼此真实身份。

API管理



Service center不同于竞品独有的API文档管理能力，可以托管系统中所有服务的API文档，各个团队成员或者管理者可以在service center中查看服务文档，并以此为设计，开发依据。微服务版本与文档为绑定关系，提高了沟通效率以及可靠性。

开发框架



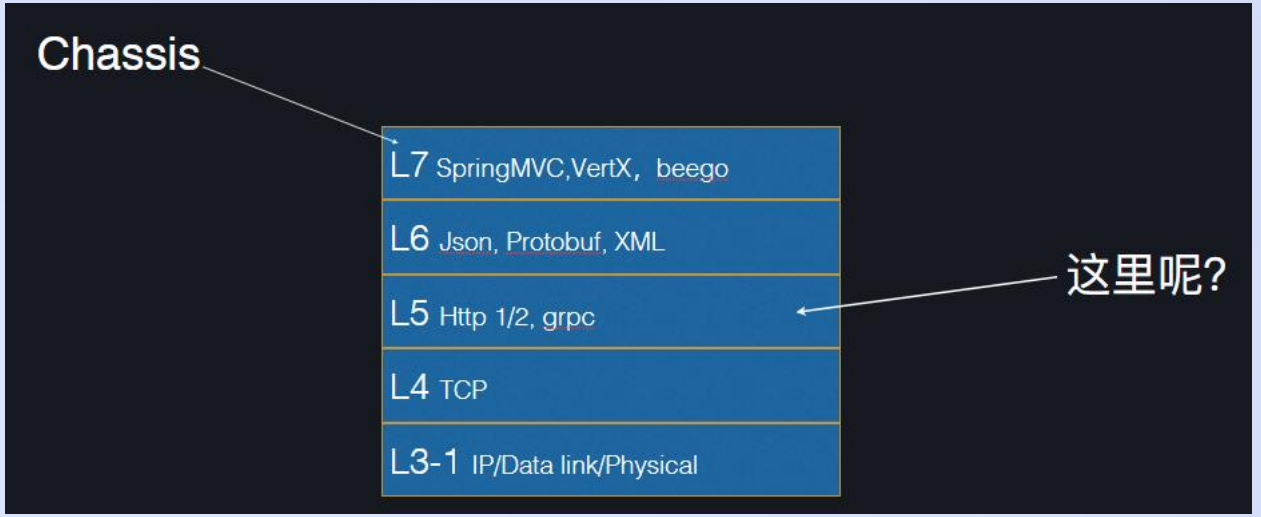
要完成以上的功能需要在微服务中编写代码，而这些代码都可以作为通用库来提供，这就是微服务开发框架。

开发者引入框架并学习开发方式，配置方式，就可以快速开发出具备微服务特性的应用。

我们称这种模式为chassis，华为云提供java-chassis与go-chassis 2种语言框架，供用户选择

Day2-14课程将介绍如何使用框架进行开发。

Service Mesh

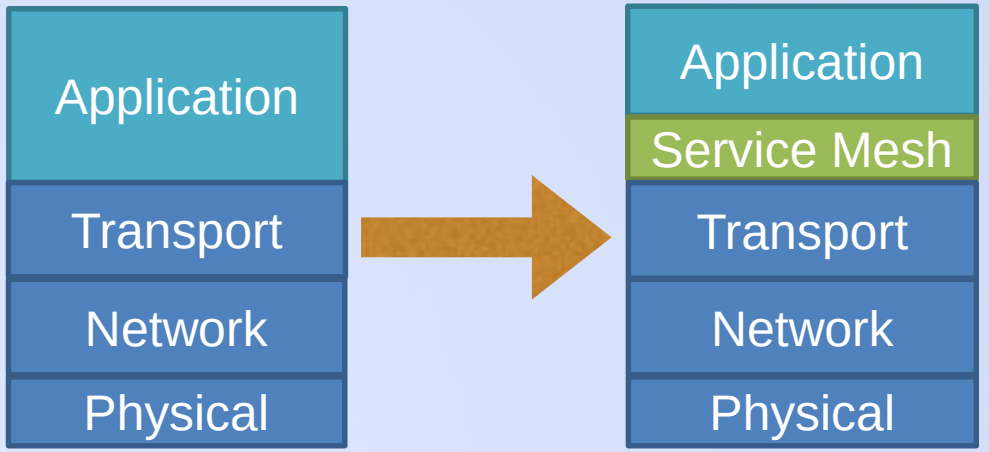


Service Mesh同样解决微服务面临的问题，但是以另一种结题思路完成的

我们在第7层使用各种开发框架，如传统开发框架Spring MVC， 微服务开发框架java-chassis，go-chassis等等，编码时直接调用。

那么微服务开发框架的功能下沉到5层呢

Service Mesh



2017年由William Morgan提出，一种基础设施层，服务间通信通过Service Mesh进行，一种TCP/IP之上的网络模型，轻量网络代理，与业务部署在一起，负责可靠传输复杂拓扑网络中的请求，将应用变为现代的云原生应用。

可以简单地理解为以前应用跑在TCP/IP之上，现在跑在Service mesh之上以处理微服务模式带来的问题。

Service Mesh的专题将在Day6的直播中介绍。

微服务平台

除了开发，微服务还需要解决的是将持续集成，基础设施，监控，中间件等功能组合在一起的平台为开发者提供良好的服务，这部分将在Day15-21介绍

Thank You