答辩稿

**开场白：**

各位评委老师，下午好！我是来自微服务产品二组的后台研发，我叫耿协锋，本次晋升高级1。

**目录：**

整个答辩内容围绕以下4点进行阐述。

**个人成长经历：**

第一部分是个人成长经历，

15年本科毕业于华东理工大学，

…

21加入腾讯云科技（武汉团队）...

**架构简介：**

首先，简单介绍一下TSF，

TSF是一个围绕应用和微服务的paas平台，其核心功能主要包括：

1、兼容多种开源微服务框架（Spring Cloud、Dubbo、ServiceMesh等）；

2、还支持API级别的服务治理能力，包括服务注册与发现、服务鉴权、服务限流、服务编排等基础服务治理能力；

3、以及devops开发与运维一体化；

4、同时还提供多维度的数据化运营，包括业务日志、服务依赖拓扑图、调用链详情、监控告警等多项应用监控数据。

TSF主要分为5个部分…，其依托于tcs底座部署运行，依赖其网络环境、MariaDB、容器服务等中间件产品。

**背景:**

前面简单介绍了下TSF，现在我们看下TSF跟TCS又有什么区别跟联系，这里拿TSF独立版来做对比：

首先，TSF缺乏更成熟的产品级高可用方案，比如支持多AZ+仲裁、自动化容灾等；

其次，运维成本高，缺少高效的运维工具和手段，比如tad、kaleido、cobra等工具，

以及交付管理这块的欠缺，比如极光（客户项目交付管理）、安灯（一线工单直接闭环掉）；

再就是，监控告警能力薄弱以及日志服务单一，只提供简单的日志检索功能；

最后是，与IAAS层兼容性差，客户最不希望被某个厂商绑架，希望能方便集成第三方主流IAAS产品。

以上能力的缺失，正是TSF拥抱TCS的主要因素。

**重点工作概述：**

下面正式进入第2点，重点工作概述，

我会从2个专项建设来阐述关键工作成果：

其中第一个专项为打通与TCS平台容器集群的纳管能力，这里主要解决的开发任务为：123

第二个专项为主导完成TCS230版本的开发运维工作,

这块的挑战还是蛮大的，时间紧、任务重（主要是给某客户承诺过交付时间，留给研发、测试的时间只有一个月左右）。

这块还有一个主要工作成果是双版本的兼容适配工作，支持x86、arm双版本交付。

按时交付上汽、福彩、中行等客户，并支持其顺利完成升级工作。

**打通与TCS平台容器集群纳管能力：**

首先我们来看第一部分的内容，打通与TCS平台容器集群纳管能力，

先来看下客户的一个痛点场景：

我们的tcs云产品主要面对的是大中型企业，一般小企业是没有能力支撑起这样一套云产品的，在大中型企业中可能已经有技术人员搭建过一套第三方集群在维护，那么此时又采购了我们的TCS云产品，但只能在我们的云产品中进行新建集群，已存在的第三方集群则需要额外维护，增加运维成本。

解决方案,将已存在的第三方集群纳管至TSF-PAAS平台，进行集中管理。

由tsf控制台发起创建集群的请求，tsf提供2中创建集群的方式，

一种是调用tcs容器平台创建集群的接口，

一种是调用tcs容器平台获取第三方纳管集群认证授权信息接口，将第三方集群成功纳管至TSF-PAAS平台。

**分析于思考：**

首先，我们进行一个现状分析：结合tsf现有架构，请求发送至tsf-resource组件，左边是虚机集群管理相关操作，主要看右边容器集群相关操作，

分为2部分，一部分是集群相关操作，比如创建集群、获取纳管集群认证信息等，通过tsf-adapter组件对接tcs容器平台接口实现；

一部分为部署相关操作，成功纳管至tsf的容器集群，通过k8s api部署应用。

那么具体如何实现，这里带来3个问题：123

**获取K8S认证授权信息-具体实现：**

首先，来看下获取K8S认证授权信息的具体实现，tsf控制台发起请求到resource组件，resource调用tsf-adapter组件，

tsf-adapter组件组装请求参数调用tcs容器平台接口，获取k8s集群的kubeconfig信息并返回。

这里的kubeconfig信息主要包含k8s集群的ca根证书以及集群context上下文信息，还包含集群的apiserver地址，以及集群所属用户的证书跟私钥。

resource组件获取到集群的kubeconfig信息后，进行k8s 客户端初始化，然后分装k8s api接口操作k8s集群。

右边是关键代码的具体实现。

以上步骤完成之后，就可以成功的将TCS容器集群纳管至TSF进行管控。

**如何打通TCS镜像仓库网络、认证？**

前面已经成功将TCS容器集群导入TSF进行管控，但是在tsf管控平台创建部署组时，发现镜像拉取不到，

通过现状分析，还需要打通tcs镜像仓库与第三方k8s集群网络，以及私有仓库认证问题，

具体解法：

首先，打通镜像网络，有2种方式，方式一、方式二；

其次，配置docker镜像仓库地址；

然后，还需要配置私有镜像仓库认证；

最后，重启docker和kubelet；

发现镜像正常拉取，服务正常运行。至此，才算成功打通与TCS容器集群的纳管能力。

**获取集群节点信息接口耗时-性能优化**：

前面将tcs容器集群导入TSF进行管控之后，发现获取集群节点信息接口查询比较耗时，特别是在集群节点比较多的时候体现比较明显；

因此，需对其性能进行优化。

首先，我们看下优化三步曲：

精简流程，减少请求资源消耗

添加缓存，减少数据库连接

并发执行，使用多线程、异步、future等编程技术，降低接口耗时等待

优化前跟优化后对比，性能提升将尽7倍

**TCS230-TSF1.29.8开发运维支撑-从挑战开始：**

ok，下面开始阐述第二个专项内容，这部分任务的挑战还是蛮多的，

首先对接tcs底座产品接口信息不同步，通过腾讯会议多次对齐，

再就是时间紧、任务重，一个月时间完成涉及30+组件改动，

缺乏人力，因tcs环境比较复杂，新人难以快速上手，临时抽调也会增加沟通成本。

环境不稳定，各产品使用同一套联调环境，难免相互影响，资源紧张的情况出现，后面申请搭建tsf自有环境维护，隔离外部干扰；

流程复杂，通过流水线建设，标准化工作流程，减少中间出错环节，提高提测质量，缩短交付时间，最终完成按时交付，并得到福彩、中行等客户的感谢信，这个后面能力评价会有相关举证。

**流水线建设-工作流程标准化：**

首先，我们通过tapd单来管理和跟踪研发过程，任务主要来自产品需求，以及bug单，其中bug单主要来自内部测试以及客户工单，

通过qci2.0流水线来持续集成、部署环境、测试、合流出包、到交付，整个流程除了测试需要人工介入，其余全部自动化完成；

**这里的主要举证成果为：**

从0到1完成2.0流水线构建；

以及完成TED弹性交付规范对接，支持x86、arm双架构脚本适配；

标准化之前流程混乱，发布时间长，标准化之后提测质量好，交付时间变短，这里有得PM以及tsf测试负责人的好评；

流水线构建脚本优化从原来的6分多缩减到1分多。

**双版本架构弹性交付的具体实现**：

再来介绍下双版本架构弹性交付的具体实现，

主要涉及5个部分，重点介绍下镜像的实现，一共涉及39个组件需要容器化处理，

主要需要适配的工作内容涉及动态配置，通过tcs环境渲染客制化配置，

以及软件包的构建，通过2.0流水线构建x86、arm软件包，及其依赖的软件包，

镜像构建，镜像是通过 dockerfile 构建的，此文件需要手动编写，符合标准的Dockerfile语法。

shell脚本编写，启动、健康检查等脚本适配。

这块的对接实现，也是tcs230版本的一个重要产出成果。

**TCS-TSF开发运维支撑-方法论沉淀：**

最后，方法论总结；

首先，日常工作中，规范开发流程，加强接口自测，建立值班制度；

其次，通过服务监控、云哨告警等机制，及时发现问题；

然后，建立完善的快速恢复策略，有变更，先回滚，一键恢复脚本等机制

最后，问题复盘，主要从根因分析、补救方案，改进落地3个方面进行问题复盘。

形成一套及时发现问题、分析问题、解决问题 内部直接闭环的方法理论。

腾讯云TCS

一是拥有多种 IaaS 适配能力，不仅能构建在自己 IaaS 产品之上，还能与开源的 openstack、VmWare、其他的云厂商 IaaS 产品平滑兼容；

二是多种 OS 适配能力，腾讯云TCS 已兼容 Tencent OS 和 CentOS 等系统；

最后是硬件适配能力，已适配海光、鲲鹏、飞腾等多种 ARM 和 X86的 CPU。

云原生是基于分布部署和统一运管的分布式云，以容器、微服务、DevOps等技术为基础建立的一套云技术产品体系。

方案特点

与底层解耦，兼容异构基础设施，用户可轻松实现基于容器的 PaaS 云平台。

腾讯云成熟 PaaS 产品+微服务框架+ DevOps 产品组合，实现业务系统快速上云。

领先的云原生应用编排与弹性伸缩能力，轻松扩容，从容应对业务高峰。

全高可用分布式架构，支持多可用区容灾；基于 eBPF 的网络技术，实现容器、服务间的高性能网络。

Tencent TCS 目前已整合了以下腾讯云 PaaS 产品：分布式数据库 TDSQL、TencentDB for Redis、消息队列 TDMQ、消息队列 CKafka、微服务平台 TSF、云开发 TCB、CODING DevOps、RIOGW 智能网关、KMS 等。

学习是创新的源泉，只有不断的学习，才能使我们专业技能以及综合能力得到提升。

问题复盘：讲一个例子；

之前福彩升级，由于出包流程的问题，现场拿错了组件版本，导致部署出现了各种问题，

后来我们从新梳理了一下出包流程，增加了几个check的环节（打生产tag前、研发、测试一起确认，现场增加脚本check机制）