解决方案实践

拓维智慧教育云平台解决方案实践

文档版本 01

发布日期 2023-11-27

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

安全声明

漏洞声明

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该政策可参考华为公司官方网站的网址：https://www.huawei.com/cn/psirt/vul-response-process。
如企业客户须获取漏洞信息，请访问：https://securitybulletin.huawei.com/enterprise/cn/security-advisory。

目 录

1 方案概述

应用场景

客户的痛点：

随着教育的发展，通过信息化、数据化、智能化推动教育高质量发展的要求越发迫切，教育的精准管理、教育的因材施教、个性学习、智能评价等多个维度的诉求越发需要结合应用系统、数据系统、智能化系统来实现，通过智慧教育云平台体系化的构建教育的数字底座、融合大数据、物联网、人工智能等先进技术，为教育的各层级、各参与者的需求进行适配，推动教育向高质量发展迈进。

痛点一：教育部门在信息化建设过程中，前期进行单点业务建设，造成信息化建设数据孤岛、用户多处应用登录，造成应用不便捷，数据无法互通。通过基础平台构建，打通统一认证能力，通过数据治理，形成数据共享服务。支撑应用的便捷使用，数据的互联互通。

痛点二：现有系统数据价值难以发挥，需要建立统一的数据标准，形成教育数据资产，为教育的精准决策提供数据支撑。提供基于国家基础教育信息标准，体系化构建教育资产目录，通过多样数据采集途径，一数一源，构建教育数据资产，为教育精准决策提供数据支撑。

痛点三： 区域建设面临应用汇聚、资源汇聚、数据汇聚的需求，希望通过统一的平台为各级教育管理员、教师、学生提供服务。通过智慧教育云平台汇聚应用、资源、数据，通过千人前面门户为各类人员提供相应服务。

痛点四： 基于数字政府要求、自研要求、安全要求，需要对信息化系统进行重构及新建。通过全栈自研产品及自研迁移能力，为教育局提供整体的数字化、自研智慧教育云平台建设。

痛点五：需要新型的技术能力，比如大数据、物联网等深度融入到教育信息化的发展中去，急需对教育的底层能力进行提升。以大数据、物联、区块链、AI等技术构建新型教育信息化底座，支撑教育业务的数字化转型。

痛点六：符合国家规范的解决方案，满足国家等保、分保、国密的规定，保障组织机构的安全运营。

通过本方案实现的业务效果：

智慧教育云平台围绕教育数字化转型，提供局校一体的数字化底座、业务平台、统一门户，为区域教育的智能治理、智慧教学、个性学习、智慧评价进行赋能。

通过大数据、AI、物联网、区块链等技术的深度融合，构建支撑区域教育数字化转型发展的数字化底座，支撑教育的学校管理、人事管理、教育教学、教育考试、教育督

导、教育教研、学生学习、发展评价等各项教育业务的数字化转型发展，通过纵向业务层级数据贯通，横向业务数据融通，体系化的构建教育数字化新范式。

智慧教育云平台从局校一体构建，主要包含以下产品内容：

区域教育数字化底座（客户：教育局信息中心）

基础支撑平台：教育基础支撑平台，支持学生成长、教师发展、教育管理核心场景构建，各类应用随需而生。平台通过移动互联、区块链等技术优化管理引擎，具备技术开放、标准规范的平台接口，便捷接入软硬件环境来保障教育信息分层传递和教育场景动态实现，让教育平台成为有机整体，体现出育人功能。平台最终夯实教育信息化的基石，成为集成商应用整合的利器。

教育大数据平台：平台解决垂直应用与纵向应用系统之间数据“条块分割，标准各异”的问题， 通过数据治理能提升数据利用价值，促进大数据工具与教育管理、教学活动、诊断评价相结合，帮助用户挖掘诱因、规范办学、提升教育质量、提高服务效能。平台沉淀数据资产，方便科室管理与分析业务数据，形成“数入一源，数出一门”格局。

统一服务门户：门户作为各级用户登录平台统一入口，具备风格整体化一、聚合教育信息、展示应用特色、宣传教育工作成果功能。个人门户更注重专属定制，以服务卡片的形式区分管理者、教师、学生、家长、公众所涉及信息。支持IPV6，多端兼容，一次发布，处处能用。

教育数据治理与增值服务

局领导驾驶舱（客户：教育局信息中心）：驾驶舱作为局领导、处（科）室针对人事管理、招生管理、教师工作、经费管理、资产管理、学业分析、双减工作、素质教育、教育改革等工作进行挂图作战。实现驾驶舱指标管理、模型研制、数据来源接入和多维数据计算，通过低代码形式完成仪表盘动态组装，实现所见即所得驾驶舱设计与使用体验。

学生成长档案（客户：教育局信息中心）：通过对中小学学生基础数据、学业数据、体质数据、荣誉数据、综合评价数据的汇聚，形成6-18岁贯通式成长档案，为教育局与学校提供丰富的学生发展数据，提供便捷的学生档案在线查看。

教师发展档案(客户：教育局信息中心)：通过对区域教师基础信息、专业发展、职业发展、年度考核、师德考核、荣誉等各类教师数据汇聚，构建教师发展档案，为教育局与学校提供教师发展数据，为教师群体精细化管理提供数据支撑。

教育服务地图（客户：教育局信息中心）：面向社会提供地区公办中小学教育资源分布情况。游客可以查看到小学、初中、高中、九年一贯制、完全中学学校在地图位置，学校概括信息、教师基本信息、学生基本信息、办学条件信息。

教师职业生涯发展（客户：教育局人事科）：以管理为重点，以专业为核心，以服务为宗旨的建设方针，内容包括全域数据可信教师信息库、教师评价、骨干引领。主要目的是推进教师职业能力评价改革，开拓教师精准评价与引入的新路径，支持教师工作决策，优化教师管理，教师个人发展进行目标引导，为国家教师管理服务改革带来新的视角和建设途径。

数据智能填报系统（客户：教育局信息中心）：数据统一填报系统是专门针对教育局对学校进行数据收集场景而提供的数据采集应用，提供的从填报模板管理、填报过程监控、数据采集结果查询分析为一体的数据采集服务，满足科室日常数据统一、规范采集需求、降低学校重复上报数据问题、提升数据使用效能。

教师招聘系统（客户：教育局人事科）

实现教师招聘工作数字化，具备招聘信息发布、考生报名、信息筛选、信息审核功能。系统提供社会通道和名优通道，审核流程支持定制，招聘过程透明，促进招聘工作规范高效。

一站式教育督导与评价（客户：教育局督导科）

教育督导系统：系统以“发展性督导评估”理论为教育发展评价基础，以“以评促建”为目标，既满足定性评价、定量评价的要求，也可满足定性+定量评价的要求，系统应用于综合督导、挂牌督导、专项督导、过程性督导。实现督导指标、数据采集、督导报告业务闭环。系统广泛服务于学前教育、义务教育、普通高中、职业教育及成人教育等。

学校年度考核系统：系统以科室协同、局校协同方式，提升考核效率，实现年度考核材料统一上报，考核评分科室分工协同，学校及时查看，疑问反馈，快速汇总考核数据，生成考核排名，极大降低考核工作量，有效推动考核工作信息化、数字化。

荣誉颁证管理系统：为实现区域教育学校发展的整体情况反映，通过统一的电子荣誉证书管理，实现对区域内各学段学校荣获获取情况的信息收集，同时支持区域在线荣誉证书颁发存档，支持区域从荣誉发展维度反映各校教育发展状况。

教师研训一体平台（客户：教育局教科研中心）

教研系统：为促进区域教育均衡，提升教师课程设计与授课专业能力，系统提供名师工作室、在线集体备课、听评课、教学研讨、磨课、教研直录播、教研资源中心等功能。

互联网大赛系统：激励个人专业能力广泛提升，满足教研部门举办教师专业能力大赛需要，系统实现大赛活动组织、资源上报、专家遴选、线上审批功能，支持灵活定制评审量表。

资源征集与评优系统：激励资源共建共享，形成优质资源库，满足教研部门、名师工作室开展资源征集评优活动，系统实现征集活动灵活配置、选手推荐、资源上报、专家遴选、量表配置、匿名评审功能。将缩减活动组织周期。

教学直录播系统：系统从满足“平战结合”实际需求出发，既支持小班制教学研讨、视频会议等工作，也支持防疫期间实施较大规模线上授课，具备白板和学科工具，具有高压缩传输、低时延特点。

课堂AI分析系统：系统以“人工智能助推教师队伍建设”为建设理念，对接直录播设备，对课堂教学的音视频文件进行教学特征分析，从教学效果、课堂互动情况、教师课堂教学基本特征等方面进行分析，输出数据报告，对精准教研和教学指导提供数据支撑。

三个课堂：“专递课堂”实现跨地域、跨校同画面、同频上课；“名师课堂”实现直播和点播评课，沉积名师课程资源；“名校网络课堂”实现青年教师联合培养、教师课堂教学、“三新”改革研讨、生涯规划指导。有效弥合校际之间数字鸿沟，补地区教育短板。

校园管理系统（客户：教育局/学校信息中心）

打通学校、家庭之间的应用壁垒，拓展网络空间，构建面向师生、家长和社会大众等不同需求的人性化公共服务，涵盖公文流转、工作汇报、请假管理、设备报修、工资管理、场馆使用、物资申报、选课管理、学生选课、排课管理、社团管理、问卷调查、投票管理、班纪班风、学情分析、活动报名、活动打卡、周工作、学生日常表现评价、红领巾、学生实践等功能，支持功能定制与引入，打造完整统一、覆盖全面、应用深入、高效稳定的信息化系统。

学生德育系统（客户：教育局/学校信息中心）

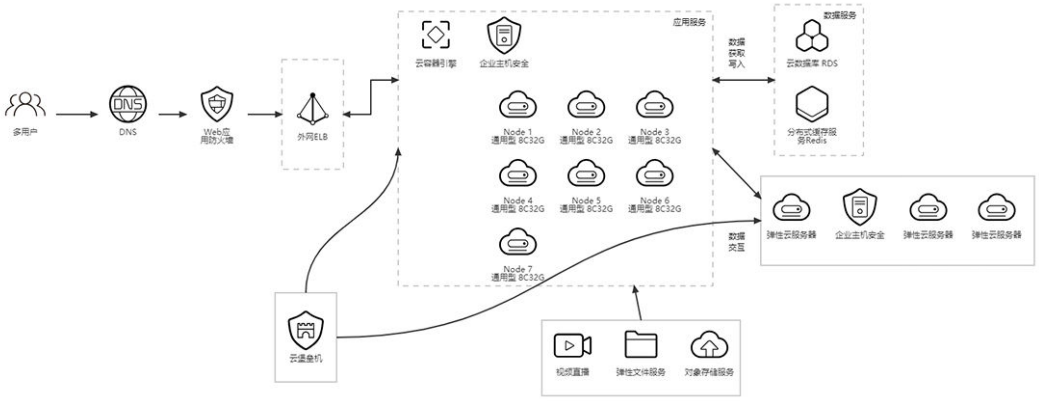
系统坚持五育并举，以活动为载体，促进班主任、科任老师、家长参与活动，减轻教师在数据评价方面工作压力，发挥家校社协同育人，让学生在阶梯式成长激励中增强少先队员光荣感。

解决方案实践的应用行业推荐：

本方案适合在省、市、区县教育局进行售卖，为区域教育构建整体智慧教育云平台。

方案架构

利用云计算、大数据、人工智能、等新技术，全面升级智慧教育云平台，建设统一身份认证系统、资源中心、开放平台、统一智慧门户，全面提升教育云一站式服务和一体化支撑能力，彻底解决平台规模化、个性化服务问题，实现便捷使用、精准服务、智能管理和开放共享，为全国“互联网+教育”大平台建设提供经验和模式。



方案优势

本方案从具备以下几个优势：

产品优势：

局校一体建设：本方案从局校整体做设计，既考虑区域共性需求，又兼容学校个性发展。

业务汇聚、数据汇聚：破除数据孤岛，通过统一的业务数据接入标准，实现教育业务数据的自动化汇聚。

千人千面门户：通过AI技术使用，让每个使用者都能获得贴合自己使用的应用环境、数据服务。

技术优势：

云原生：以云原生技术为主的底层技术支持，是集微服务、容器化、DevOps、多云适配与多环境支持为一体的综合技术支撑平台。针对系统基座支撑能力提升+业务系统瘦身的营销场景，提升对大型组织高并发、大数据、复杂业务的支撑力；对系统进行统一模型改造，实现技术底层逻辑互通；解耦复杂系统，实现最小颗粒度的组件复用。

大数据平台：依据教育数据标准构建集数据采集、治理、分析与一体的教育数据治理平台，通过模块化、流程化的设计，让教育数据资产能便捷的归集、标准化的沉淀、安全的使用，通过数据应用分析套件，实现数据的可视化展示、自定义分析与查询、数据报告的自定义生成。为教育数据的增值服务提供支撑。

低代码定制平台：面向业务人员，提供一套完整的应用开发平台，实现包括需求、开发、测试、发布、实施、运维的应用全生命周期管理，同时实现一对一专属定制和标

准应用产品的定制模式，充分融合致远在协同管理的实践与中台引擎能力，成为政企协同服务及管理创新与实践的加速器。

自研适配：全栈软件平台支持自研环境部署，公司具有自研适配中心，有整套自研迁移技术，能满足教育单位整体自研的要求。

2 资源和成本规划

本节介绍解决方案实践中资源规划情况，包含以下内容：

资源和成本规划内容说明

维度	说明
资源规划	云计算资源见《 表2-2 资源和成本规划 》 涉及软件： Mysql: 5.7 Redis: 6.2 Mongo: 4.4 Kafka: 2.5.12 Zookeeper: 3.6 Nginx: 1.24 Fastdfs: 6.06 Activemq: 5.17.6 Kkfileview: 4.1.0 Minio: 最新版 1. server：3.3.1 Jdk： 1.8
成本规划	成本规划见《 表2-2 资源和成本规划 》

资源和成本规划：

表 2-1 资源和成本规划

云资源	规格	数量
VPC	网段选择172.16.0.0/16，其他采用默认配置	1
Subnet	网段选择172.16.0.0/24，其他采用默认配置	1

云资源	规格	数量
安全组	根据需要开通入方向3306、6379等端口	1
ECS	通用计算增强型 | c6.2xlarge.4 | 8vCPUs | 32GiB 系统盘 200G数据盘	3
CCE容器引擎	CCE Standard集群，非高可用，50个节点	1
CCE 节点	通用计算增强型 | c3.4xlarge.4 | 16 vCPUs | 64 GiB | 可用区 1	6
OBS	标准存储单AZ存储包 | 5TB	1
RDS	rds.mysql.x1.2xlarge.2 | 8 vCPUs | 16 GB(独享型)	1
NAT网关	小型	1
ELB负载均衡	共享型	1
EIP弹性公网IP	100M按需	2

3 实施步骤

3.1 一、 服务器初始化

3.1.1 服务器规划及功能清单

序号	IP	主机名	部署描述	CPU	内存	数据盘
1	192.168.1.61	k8s-master1	k8s管理节点	8	16	200
2	192.168.1.195	k8s-master2	k8s管理节点	8	16	200
3	192.168.1.198	k8s-master3	k8s管理节点	8	16	200
4	192.168.1.207	k8s-node01	k8s工作节点	16	64	200
5	192.168.1.237	k8s-node02	k8s工作节点	16	64	200
6	192.168.1.235	k8s-node03	k8s工作节点	16	64	200
7	192.168.1.199	k8s-node04	k8s工作节点	16	64	200
8	192.168.1.191	k8s-node05	k8s工作节点	16	64	200
9	192.168.1.189	k8s-node06	k8s工作节点	16	64	200
10	192.168.1.57	k8s-node07	k8s工作节点	8	32	200
11	192.168.1.133	k8s-node08	k8s工作节点	8	32	200
12	192.168.1.127	k8s-node09	k8s工作节点	16	32	100
13	192.168.1.82	k8s-node10	k8s工作节点	16	32	100
14	192.168.1.156	k8s-node11	k8s工作节点	16	32	100

序号	IP	主机名	部署描述	CPU	内存	数据盘
15	192.168.1.112	zk-01	Zookeeper	8	32	100
16	192.168.1.56	zk-02	Zookeeper	8	32	100
17	192.168.1.250	zk-03	Zookeeper	8	32	100
18	192.168.1.98	Mongodb	Mongodb	8	32	100
19	192.168.1.99	R	/	8	32	100
20	192.168.1.132	/	/	8	32	100
21	192.168.1.3	harbor、rancher	harbor 仓库	8	32	100
22	192.168.1.219	基础平台mysql	基础平台mysql	8	32	100
23	192.168.1.102	智慧校mysql	智慧校mysql	8	32	100
24	192.168.1.18	学习中心mysql	学习中心mysql	8	32	100
25	192.168.1.23	大数据mysql	大数据mysql	8	32	100
26	192.168.1.148	前置mysql	前置mysql	8	32	100

3.1.2 准备 ansible 的 hosts 文件

后续批量操作使用ansible操作，以增加便捷性和安全性

3.1.3 修改服务器主机名

使用ansible批量修改服务器主机名

3.1.3.1 更改主机名脚本

- hosts: all

remote_user: root

tasks:

- name: change name ##永久修改，重启服务器后生效，主机名来自于ansible的hosts文件

shell: "echo {{ hostname|quote }} >/etc/hostname"

- name: ##临时修改，重新登录生效

shell: hostname {{ hostname|quote }}

- name: create data if not exits ##创建工作目录

file: path=/data/ state=directory

3.1.3.2 使用 ansible 更新服务器主机名

ansible-playbook hostnameChange.yaml ##批量执行修改

3.1.3.3 验证

ansible all -m shell -a "hostname"

```
k8s-node01
192.168.1.237 | CHANGED | rc=0 >>
k8s-node02
192.168.1.199 | CHANGED | rc=0 >>
k8s-node04
192.168.1.235 | CHANGED | rc=0 >>
k8s-node03
192.168.1.189 | CHANGED | rc=0 >>
k8s-node06
192.168.1.191 | CHANGED | rc=0 >>
k8s-node05
192.168.1.57 | CHANGED | rc=0 >>
k8s-node07
192.168.1.133 | CHANGED | rc=0 >>
k8s-node08
192.168.1.61 | CHANGED | rc=0 >>
k8smaster1
192.168.1.127 | CHANGED | rc=0 >>
k8s-node09
192.168.1.156 | CHANGED | rc=0 >>
k8s-node11
192.168.1.82 | CHANGED | rc=0 >>
k8s-node10
```

Figure 1:

3.1.4 安装必要软件

3.1.4.1 脚本

- name: insatll docker-ce package ##配置镜像源为内网镜像仓库

hosts: k8s

tasks:

- name: scp install packages

unarchive: src="/data/src/dockerkuber_yum.tar.gz dest=/data/"

- name: install

shell: yum -y install /data/dockerkuber_yum/*

3.1.4.2 执行脚本

ansible-playbook yumupdate.yaml

3.1.5 关闭防火墙、selinux

3.1.5.1 脚本

```
---
- name: stop firewalld selinux
hosts: all
tasks:
- name: stop firewalld service ##关闭firewalld
service: name=firewalld state=stopped enabled=no
ignore_errors: Ture
- name: stop selinux
shell: sed -i "s@SELINUX=.*@SELINUX=disabled@" /etc/selinux/config
ignore_errors: Ture
```

3.1.5.2 执行脚本

```
ansible-playbook stopSelinux.yaml
```

3.1.6 升级内核

3.1.6.1 脚本

```
---
- name: update kernel to 4.18 ##内核升级到4.18版本，内核软件包在部署文件目录内
hosts: all
tasks:
- name: copy kernel tar
unarchive: src=/etc/ansible/test.tar.gz dest=/data/
- name: install kernel
shell: yum -y install /data/test/*
- name: set default kernel
shell: grub2-set-default 0 && grub2-mkconfig -o /etc/grub2.cfg
```

3.1.6.2 执行

```
ansible-playbook kernelUpdate.yaml
```


3.1.6.3 重启服务器

为保障安全，此步骤一台台服务器单独执行,执行命令为:

shutdown -r now

3.1.6.4 重启后验证

ansible all -m shell -a "uname -r"

```
4.18.9-1.el7.elrepo.x86_64
192.168.1.250 | CHANGED | rc=0 >>
4.18.9-1.el7.elrepo.x86_64
192.168.1.112 | CHANGED | rc=0 >>
4.18.9-1.el7.elrepo.x86_64
192.168.1.132 | CHANGED | rc=0 >>
4.18.9-1.el7.elrepo.x86_64
192.168.1.219 | CHANGED | rc=0 >>
4.18.9-1.el7.elrepo.x86_64
192.168.1.3 | CHANGED | rc=0 >>
4.18.9-1.el7.elrepo.x86_64
192.168.1.18 | CHANGED | rc=0 >>
4.18.9-1.el7.elrepo.x86_64
192.168.1.102 | CHANGED | rc=0 >>
4.18.9-1.el7.elrepo.x86_64
192.168.1.23 | CHANGED | rc=0 >>
4.18.9-1.el7.elrepo.x86_64
192.168.1.198 | CHANGED | rc=0 >>
4.18.9-1.el7.elrepo.x86_64
192.168.1.195 | CHANGED | rc=0 >>
4.18.9-1.el7.elrepo.x86_64
192.168.1.148 | CHANGED | rc=0 >>
4.18.9-1.el7.elrepo.x86_64
```

Figure 2:

3.2 二、 搭建 harbor 仓库

3.2.1 服务器规划

序号	IP	主机名	部署描述
1	192.168.1.3	base02	harbor

3.2.2 获取 harbor 部署脚本

wget https://github.com/goharbor/harbor/releases/download/v2.3.1/harbor-offline-installer-v2.3.1.tgz

3.2.3 修改配置文件，开启 ssl 认证

cat harbar.yaml

hostname: harbor.talkedu.com


```
http:
port: 8080
https:
port: 443
certificate: /data/harbor/ssl/harbor.talkedu.com.cert
private_key: /data/harbor/ssl/harbor.talkedu.com.key
harbor_admin_password: Talkedu@123
database:
password: Talkedu@123
max_idle_conns: 100
max_open_conns: 900
data_volume: /data/harbor
trivy:
ignore_unfixed: false
skip_update: false
insecure: false
jobservice:
max_job_workers: 10
notification:
webhook_job_max_retry: 10
chart:
absolute_url: disabled
log:
level: info
local:
rotate_count: 50
rotate_size: 200M
location: /var/log/harbor
_version: 2.3.0
proxy:
http_proxy:
https_proxy:
no_proxy:
```


components:

- core

- jobservice

- trivy

3.2.4 创建 ssl 证书

##创建自签证书命令如下

]# openssl genrsa -out ca.key 4096

]# openssl req -x509 -new -nodes -sha512 -days 3650 -subj "/C=CN/ST=xian/L=xian/O=example/OU=Personal/CN=harbor.talkedu.com" -key ca.key -out ca.crt

]# openssl genrsa -out harbor.talkedu.com.key 4096

]# openssl req -sha512 -new -subj "/C=CN/ST=xian/L=xian/O=example/OU=Personal/CN=harbor.talkedu.com" -key harbor.talkedu.com.key -out harbor.talkedu.com.csr

]# cat >v3.ext<<-EOF

authorityKeyIdentifier=keyid,issuer

basicConstraints=CA:FALSE

keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment

extendedKeyUsage = serverAuth

subjectAltName = @alt_names

[alt_names]

DNS.1=harbor.talkedu.com

DNS.2=harbor.talkedu.com

DNS.3=base02

EOF

]# openssl x509 -req -sha512 -days 3650 -extfile v3.ext -CA ca.crt -CAkey ca.key -CAcreateserial -in harbor.talkedu.com.csr -out harbor.talkedu.com.crt

]# openssl x509 -inform PEM -in harbor.talkedu.com.crt -out harbor.talkedu.com.cert

3.2.5 执行部署脚本

bash install.sh

3.2.6 验证

docker ps -a ##所用pod均正常启动


```
[root@base02 project]# docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED       STATUS
    PORTS                                                                                 NAMES
aad43e49cfc4   goharbor/nginx-photon:v2.3.5         "nginx -g 'daemon of…"   2 hours ago   Up 2 hours (healthy
)   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 0.0.0.0:443->8443/tcp, :::443->8443/tcp   nginx
7f5e43b47a74   goharbor/harbor-jobservice:v2.3.5    "/harbor/entrypoint.…"   2 hours ago   Up 2 hours (healthy
)                                                                                         harbor-jobservice
36231c4bed42   goharbor/harbor-core:v2.3.5          "/harbor/entrypoint.…"   2 hours ago   Up 2 hours (healthy
)                                                                                         harbor-core
2edd98608cd5   goharbor/registry-photon:v2.3.5      "/home/harbor/entryp…"   2 hours ago   Up 2 hours (healthy
)                                                                                         registry
e194d70a0674   goharbor/harbor-portal:v2.3.5        "nginx -g 'daemon of…"   2 hours ago   Up 2 hours (healthy
)                                                                                         harbor-portal
4075e60423ba   goharbor/harbor-registryctl:v2.3.5   "/home/harbor/start.…"   2 hours ago   Up 2 hours (healthy
)                                                                                         registryctl
a58266181a71   goharbor/redis-photon:v2.3.5         "redis-server /etc/r…"   2 hours ago   Up 2 hours (healthy
)                                                                                         redis
2159f3347c43   goharbor/harbor-db:v2.3.5            "/docker-entrypoint.…"   2 hours ago   Up 2 hours (healthy
)                                                                                         harbor-db
6aa20ddfb757   goharbor/harbor-log:v2.3.5           "/bin/sh -c /usr/loc…"   2 hours ago   Up 2 hours (healthy
)   127.0.0.1:1514->10514/tcp                                                             harbor-log
[root@base02 project]#
```

Figure 3:

3.2.7 配置防火墙规则，限制 IP 访问

iptables -I INPUT -p tcp -m iprange --src-range=192.168.1.1-192.168.1.254 -m multiport --dports 443 -j ACCEPT

iptables -A INPUT -p tcp -m multiport --dport 443 -j DROP

iptables-save > /etc/sysconfig/iptables ##保存规则，服务器意外重启后能自动加载配置

3.3 三、 部署 Kubernetes 集群

3.3.1 软件版本

- 系统： Centos 7.7
- Kernel Version： 4.18.9-1.el7.elrepo.x86_64
- Kubernetes： v1.19.16
- calico: v3.19.1
- docker-ce: 20.10.12
- VIP: 内网负载均衡

3.3.2 服务器清单

序号	IP	主机名	部署描述
1	192.168.1.61	k8s-master1	k8s管理节点
2	192.168.1.195	k8s-master2	k8s管理节点
3	192.168.1.198	k8s-master3	k8s管理节点
4	192.168.1.207	k8s-node01	k8s工作节点
5	192.168.1.237	k8s-node02	k8s工作节点

序号	IP	主机名	部署描述
6	192.168.1.235	k8s-node03	k8s工作节点
7	192.168.1.199	k8s-node04	k8s工作节点
8	192.168.1.191	k8s-node05	k8s工作节点
9	192.168.1.189	k8s-node06	k8s工作节点
10	192.168.1.57	k8s-node07	k8s工作节点
11	192.168.1.133	k8s-node08	k8s工作节点
12	192.168.1.127	k8s-node09	k8s工作节点
13	192.168.1.82	k8s-node10	k8s工作节点
14	192.168.1.156	k8s-node11	k8s工作节点

3.3.3 优化 kubernetes 集群

3.3.3.1 关闭 swap

ansible k8smaster -m shell -a "swapoff -a && sed -ri 's/.*swap.*/#&/' /etc/fstab"

3.3.3.2 优化服务器内核参数

3.3.3.2.1 脚本

##优化内核参数，文件k8s.conf、limits.conf在部署文件夹内

- name: config sysctl

hosts: k8s

tasks:

- name: copy sysctl kernel argus

copy: src=/etc/ansible/k8s.conf dest=/etc/sysctl.d/

- name:

shell : sysctl --system

- name: config limit

copy: src=/etc/ansible/limits.conf dest=/etc/security/limits.conf

3.3.3.2.2 执行

ansible-playbook sysctl_config.yaml

3.3.3.2.3 验证

ansible k8smaster -m shell -a "sysctl --system"

3.3.3.3 开启 ip_vs 模块

3.3.3.3.1 脚本

##k8s集群使用ip_VS模式，能大大提高性能，减少iptables规则数量

- name: start ipvs module

hosts: k8s

tasks:

- name:

shell : modprobe -- ip_vs && modprobe -- ip_vs_rr && modprobe -- ip_vs_wrr && modprobe -- ip_vs_sh && modprobe -- nf_conntrack_ipv4

3.3.3.3.2 执行脚本

ansible-playbook ipvs.yaml

3.3.3.3.3 验证

ansible k8s -m shell -a "lsmod | grep ip_vs"

```
192.168.1.57 | CHANGED | rc=0 >>
ip_vs_sh               16384  0
ip_vs_wrr              16384  0
ip_vs_rr               16384  15
ip_vs                 151552  21 ip_vs_rr,ip_vs_sh,ip_vs_wrr
nf_conntrack          135168  7 xt_conntrack,nf_conntrack_ipv4,nf_nat,ipt_MASQUERADE,nf_nat_ipv4,nf_conntrack_netlink,ip_vs
libcrc32c              16384  3 nf_conntrack,nf_nat,ip_vs
192.168.1.61 | CHANGED | rc=0 >>
ip_vs_sh               16384  0
ip_vs_wrr              16384  0
ip_vs_rr               16384  15
ip_vs                 151552  21 ip_vs_rr,ip_vs_sh,ip_vs_wrr
nf_conntrack          135168  9 xt_conntrack,nf_conntrack_ipv6,nf_conntrack_ipv4,nf_nat,nf_nat_ipv6,ipt_MASQUERADE,nf_nat_ipv4,nf_conntrack_netlink,ip_vs
libcrc32c              16384  3 nf_conntrack,nf_nat,ip_vs
```

Figure 4:

3.3.4 启动 docker

3.3.4.1 准备启动脚本

- name: start docker-ce

hosts: k8smaster

tasks:

- name: make docker daemon directory ##创建镜像加速、配置目录

file: path=/etc/docker/ state=directory

- name: copy docker daemon.json file ##此文件内容见下方

copy: src=/etc/docker/daemon.json dest=/etc/docker/daemon.json

- name: Support docker tab key completion ##支持docker使用tab键命令补全功能

shell: source /usr/share/bash-completion/completions/docker && source /usr/share/bash-completion/bash_completion

- name: support kubeadm kubectl tab key completion ##支持kubectl、kubeadm使用tab键命令补全功能

shell: echo 'source <(kubectl completion bash)' >> /root/.bashrc \

&& source /root/.bashrc

- name: start docker ##启动docker容器

service: name=docker state=started enabled=yes

- name: enable kubelet ##kubelet开机自启，初始化kubernetes必须执行此操作

shell : systemctl enable kubelet

附： deamon.json文件内容：

```
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": { ##docker日志本地保存策略，保存500m，每个容器保存5个日志文件
"max-size": "500m",
"max-file": "5"
},
"storage-driver": "overlay2",
"storage-opts": [
"overlay2.override_kernel_check=TRUE"
],
"registry-mirrors":["https://r2hd8p9u.mirror.aliyuncs.com"],
"data-root": "/data/docker" ##使用/data/docker作为docker根路径
}
```

3.3.4.2 执行脚本

ansible-playbook startdocker.yaml

3.3.4.3 验证

ansible k8s -m shell -a "systemctl status docker "

```
192.168.1.156 | CHANGED | rc=0 >>
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disable
d)
   Active: active (running) since Thu 2022-03-31 15:58:55 CST; 1h 11min ago
     Docs: https://docs.docker.com
 Main PID: 3477 (dockerd)
    Tasks: 22
   Memory: 511.3M
   CGroup: /system.slice/docker.service
           └─3477 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Mar 31 15:58:55 k8s-node11 dockerd[3477]: time="2022-03-31T15:58:55.122548614+08:00" level
=info msg="Loading containers: start."
Mar 31 15:58:55 k8s-node11 dockerd[3477]: time="2022-03-31T15:58:55.174490069+08:00" level
=info msg="Default bridge (docker0) is assigned with an IP address 172.17.0.0/16. Daemon o
ption --bip can be used to set a preferred IP address"
Mar 31 15:58:55 k8s-node11 dockerd[3477]: time="2022-03-31T15:58:55.197116650+08:00" level
=info msg="Loading containers: done."
Mar 31 15:58:55 k8s-node11 dockerd[3477]: time="2022-03-31T15:58:55.214026544+08:00" level
=info msg="Docker daemon" commit=459d0df graphdriver(s)=overlay2 version=20.10.12
```

Figure 5:

3.3.5 集群初始化(此步骤在 master01 上执行)

3.3.5.1 准备 kubeadm 初始化文件

cat >> kubeadm-config.yaml << EOF

apiVersion: kubeadm.k8s.io/v1beta2

bootstrapTokens:

- groups:

- system:bootstrappers:kubeadm:default-node-token

token: abcdef.0123456789abcdef

ttl: 24h0m0s

usages:

- signing

- authentication

kind: InitConfiguration

localAPIEndpoint:

advertiseAddress: 192.168.1.61

bindPort: 6443

nodeRegistration:

criSocket: /var/run/dockershim.sock

name: nx-master01


```
taints:
- effect: NoSchedule
key: node-role.kubernetes.io/master
---
apiServer:
timeoutForControlPlane: 4m0s
apiVersion: kubeadm.k8s.io/v1beta2
certificatesDir: /etc/kubernetes/pki
clusterName: kubernetes
controllerManager: {}
controlPlaneEndpoint: "k8s.talkedu.com:6443" ##kube-apiserver使用IP，为内网负载均衡器IP
dns:
type: CoreDNS
etcd:
local:
dataDir: /var/lib/etcd
imageRepository: harbor.talkedu.com/system ##使用镜像仓库下载镜像
kind: ClusterConfiguration
kubernetesVersion: v1.19.16 ##部署版本为1.19.3
networking:
dnsDomain: cluster.local
serviceSubnet: "10.96.0.0/12"
podSubnet: "10.244.0.0/16" ##pod网段，此字段需与calico网段保持一致
scheduler: {}
---
apiVersion: kubeproxy.config.k8s.io/v1alpha1
kind: KubeProxyConfiguration ###集群使用ipvs作为网络模式
mode: "ipvs"
EOF
```

3.3.5.2 开始初始化 kubeadm 控制平面

```
kubeadm init --config=kubeadm-config.yaml --upload-certs
```


3.3.5.3 保存加入集群命令

将上面命令打印的kubeadm join命令保存，后续控制平面和数据平面加入集群时需要用到。

3.3.5.4 配置 kubectl 访问集群配置文件

配置kubectl命令管理kubernetes集群

mkdir /root/.kube && cp /etc/kubernetes/admin.conf /root/.kube/config

3.3.5.5 验证

kubectl get node && kubectl get pods -n kube-system

```
[root@k8smaster1 src]# kubectl get node && kubectl get pods -n kube-system
NAME         STATUS   ROLES    AGE   VERSION
k8s-node01   Ready    <none>   56m   v1.19.16
k8s-node02   Ready    <none>   55m   v1.19.16
k8s-node03   Ready    <none>   54m   v1.19.16
k8s-node04   Ready    <none>   54m   v1.19.16
k8s-node05   Ready    <none>   53m   v1.19.16
k8s-node06   Ready    <none>   53m   v1.19.16
k8s-node07   Ready    <none>   52m   v1.19.16
k8s-node08   Ready    <none>   51m   v1.19.16
k8s-node09   Ready    <none>   51m   v1.19.16
k8s-node10   Ready    <none>   50m   v1.19.16
k8s-node11   Ready    <none>   50m   v1.19.16
k8smaster1   Ready    master   67m   v1.19.16
k8smaster2   Ready    master   65m   v1.19.16
k8smaster3   Ready    master   59m   v1.19.16
```

Figure 6:

```
[root@k8smaster1 src]#  kubectl get pods -n kube-system
NAME                                       READY   STATUS    RESTARTS   AGE
calico-kube-controllers-6f497785c5-7w8mv   1/1     Running   2          61m
calico-node-6jd9x                          1/1     Running   0          50m
calico-node-6rczr                          1/1     Running   0          55m
calico-node-7gh7p                          1/1     Running   1          61m
calico-node-92h8b                          1/1     Running   0          61m
calico-node-94kdh                          1/1     Running   0          54m
calico-node-d7c22                          1/1     Running   0          51m
calico-node-dnrfw                          1/1     Running   0          59m
calico-node-fkmcq                          1/1     Running   0          52m
calico-node-gfqbr                          1/1     Running   0          51m
calico-node-hkpdt                          1/1     Running   0          56m
calico-node-hwxnb                          1/1     Running   0          53m
calico-node-p4pfw                          1/1     Running   0          54m
calico-node-pk24s                          1/1     Running   0          56m
calico-node-xrfcx                          1/1     Running   0          53m
coredns-69475cc69f-k2tl2                   1/1     Running   0          68m
coredns-69475cc69f-vnkkr                   1/1     Running   1          68m
etcd-k8smaster1                            1/1     Running   1          68m
etcd-k8smaster2                            1/1     Running   0          66m
etcd-k8smaster3                            1/1     Running   0          59m
```

Figure 7:

3.3.6 安装 calico 网络插件

3.3.6.1 准备安装清单文件

calico.yaml ##此文件在部署文件内

3.3.6.2 应用清单文件

kubectl apply -f calico.yaml

3.3.6.3 验证

kubectl get pods -n kube-system ##查看次名称空间内所有Pod是否成功启动

```
[root@k8smaster1 src]#  kubectl get pods -n kube-system
NAME                                       READY   STATUS    RESTARTS   AGE
calico-kube-controllers-6f497785c5-7w8mv   1/1     Running   2          61m
calico-node-6jd9x                          1/1     Running   0          50m
calico-node-6rczr                          1/1     Running   0          55m
calico-node-7gh7p                          1/1     Running   1          61m
calico-node-92h8b                          1/1     Running   0          61m
calico-node-94kdh                          1/1     Running   0          54m
calico-node-d7c22                          1/1     Running   0          51m
calico-node-dnrfw                          1/1     Running   0          59m
calico-node-fkmcq                          1/1     Running   0          52m
calico-node-gfqbr                          1/1     Running   0          51m
calico-node-hkpdt                          1/1     Running   0          56m
calico-node-hwxnb                          1/1     Running   0          53m
calico-node-p4pfw                          1/1     Running   0          54m
calico-node-pk24s                          1/1     Running   0          56m
calico-node-xrfcx                          1/1     Running   0          53m
coredns-69475cc69f-k2tl2                   1/1     Running   0          68m
coredns-69475cc69f-vnkkr                   1/1     Running   1          68m
etcd-k8smaster1                            1/1     Running   1          68m
etcd-k8smaster2                            1/1     Running   0          66m
etcd-k8smaster3                            1/1     Running   0          59m
```

Figure 8:

3.3.7 将其余 master 节点和 node 节点加入集群

3.3.7.1 在 master 节点上执行

kubeadm join k8s.talkedu.com:6443 --token abcdef.0123456789abcdef \

--discovery-token-ca-cert-hash sha256:1df03bef705ad54fa693121e644d372d37d6b7e79a45058a999bc2d2ff590776 \

--control-plane --certificate-key 8bbe0725b691eff7ea4e92b4ef70a797ce3224a84e8ad8c02c42e70b32de3df0

##此命令仅24小时内有效，过期需重新打印加入集群指令,执行命令为： kubeadm token create --print-join-command

3.3.7.2 在 node 节点上执行

kubeadm join k8s.talkedu.com:6443 --token abcdef.0123456789abcdef \

--discovery-token-ca-cert-hash sha256:1df03bef705ad54fa693121e644d372d37d6b7e79a45058a999bc2d2ff590776

##此命令仅24小时内有效，过期需重新打印加入集群指令，执行命令为： kubeadm token create --print-join-command

3.3.7.3 验证

kubectl get nodes && kubectl get pods -A

3.3.8 更新证书有效期 10 年

3.3.8.1 准备更新脚本

update-kubeadm-cert.sh ##此文件在部署文件内，kubernetes集群默认证书有效期为1年

3.3.8.2 执行脚本

bash update-kubeadm-cert.sh all

此脚本能更新除kubelet之外的所有证书

3.3.8.3 验证

kubeadm alpha certs check-expiration

```
[root@k8smaster1 src]# kubeadm alpha certs check-expiration
[check-expiration] Reading configuration from the cluster...
[check-expiration] FYI: You can look at this config file with 'kubectl -n kube-system get
cm kubeadm-config -oyaml'

CERTIFICATE                EXPIRES                  RESIDUAL TIME   CERTIFICATE AUTHORITY
  EXTERNALLY MANAGED
admin.conf                 Mar 28, 2032 08:22 UTC   9y
  no
apiserver                  Mar 28, 2032 08:22 UTC   9y              ca
  no
apiserver-etcd-client      Mar 28, 2032 08:22 UTC   9y              etcd-ca
  no
apiserver-kubelet-client   Mar 28, 2032 08:22 UTC   9y              ca
  no
controller-manager.conf    Mar 28, 2032 08:22 UTC   9y
  no
etcd-healthcheck-client    Mar 28, 2032 08:22 UTC   9y              etcd-ca
  no
etcd-peer                  Mar 28, 2032 08:22 UTC   9y              etcd-ca
  no
```

Figure 9:

3.3.8.4 升级 kubelet 证书

##在kube-controller-manager配置文件中，在command内添加最后两行，配置更新证书期限为10年，并开启证书自动轮转

vim /etc/kubernetes/manifests/kube-controller-manager.yaml

- command:

- kube-controller-manager

- --experimental-cluster-signing-duration=87600h0m0s

- --feature-gates=RotateKubeletServerCertificate=TRUE

重启kubelet服务

systemctl restart kubelet

3.3.9 部署 ingress-nginx

提供对外访问入口，将此服务的80、443端口映射到主机上，访问时，通过该80、443端口，访问集群内所有服务。

3.3.9.1 准备安装清单文件

ingress-nginx.yaml ##此文件在部署文件内

3.3.9.2 应用清单文件

kubectl apply -f ingress-nginx.yaml

3.3.9.3 验证

kubectl get pods -n ingress-nginx

```
[root@k8smaster1 src]# kubectl  get pods -n ingress-nginx
NAME                                   READY   STATUS      RESTARTS   AGE
ingress-nginx-admission-create-xc27r   0/1     Completed   0          39m
ingress-nginx-admission-patch-z26zj    0/1     Completed   1          39m
ingress-nginx-controller-76f6c4fdfb-7n4gn   1/1     Running     0          39m
```

Figure 10:

4 修订记录

发布日期	修订记录
2023-10-31	第一次正式发布。