云应用引擎

最佳实践

文档版本01发布日期2025-01-22





版权所有 © 华为云计算技术有限公司 2025。保留一切权利。

非经本公司书面许可,任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部,并不得以任何形式传播。

商标声明

NUAWE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云计算技术有限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,华为云计算技术有限公司对本文 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。

华为云计算技术有限公司

地址: 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编: 550029

网址: <u>https://www.huaweicloud.com/</u>

| _ | িয়ন্থ |
|---|--------|
| | |

| 1 CAE 最佳实践汇总 | 1 |
|---------------------------------------|----|
| 2 使用 CAE 托管 Nginx 静态文件服务器 | 3 |
| 21 概述 | 3 |
| 22 部署前准备 | |
| 2.3 操作步骤 | 8 |
| 3 Gitlab 对接 Jenkins 自动构建并部署到 CAE | 15 |
| 3.1 概述 | 15 |
| 3.2 环境准备 | 16 |
| 3.2.1 准备 Jenkins 环境 | 17 |
| 3.2.2 上传代码到 Gitlab 代码仓库 | 18 |
| 3.2.3 安装 obsutil 工具 | 19 |
| 3.2.4 安装 KooCLI 工具 | 20 |
| 3.2.5 安装插件并配置 Jenkins 工具 | 21 |
| 3.3 操作步骤 | 25 |
| 3.3.1 操作前对接测试 | 25 |
| 3.3.2 配置流水线构建任务 | 28 |
| 3.3.3 deploy.sh 脚本说明 | 34 |
| 3.4 构建验证 | 42 |
| 3.4.1 手动构建验证 | 42 |
| 3.4.2 Gitlab 自动触发 Jenkins 构建 | 43 |
| 3.5 附录 | 45 |
| 3.5.1 AK/SK 获取方法 | 45 |
| 4 Jenkins 流水线支持多组件按照依赖顺序部署 | 47 |
| 4.1 概述 | 47 |
| 4.2 前提条件 | 48 |
| 4.3 操作步骤 | 49 |
| 5 SpringCloud 应用部署到 CAE 自动对接 Nacos 引擎 | 59 |
| 5.1 概述 | 59 |
| 5.2 前提条件 | 61 |
| 5.3 操作步骤 | 61 |
| 6 Spring Cloud 应用无损上线 | 66 |

| 7 健康检查 | 68 |
|---|-----|
| 7.1 概述 | |
| 7.2 启动探针与存活探针配合工作 | |
| 7.3 使用就绪探针保证升级时流量正常 | |
| 8 生命周期管理 | 75 |
| 8.1 概述 | |
| 8.2 利用启动后处理写文件 | |
| 8.3 利用停止前处理优雅关闭 Nginx | |
| 9 发送事件告警到企业微信 | 80 |
| 10 对接软件开发生产线 CodeArts 流水线自动升级到 CAE | |
| 10.1 概述 | |
| 10.2 流水线构建软件包上传到 Codearts 软件发布库后升级 CAE 组件 | |
| 10.3 流水线构建软件包上传到 obs 桶后升级 CAE 组件 | |
| 10.4 流水线构建镜像上传到 swr 镜像仓库后升级 CAE 组件 | |
| 11 通过配置 PromQL 实现自定义弹性伸缩 | 120 |
| 12 配置 CAE 对接 DEW,帮助应用从 DEW 获取加密凭据 | 121 |
| 12.1 概述 | |
| 12.2 配置凭据,通过环境变量注入 | 122 |
| 12.3 屏蔽子账户读取密钥的权限,实现密钥保护 | 125 |
| 13 ASP.NET Core 应用部署到 CAE | 130 |
| 14 通过企业路由打通网络 | |

CAE 最佳实践汇总

本文汇总了基于云应用引擎服务(CAE,Cloud Application Engine)常见应用场景的 操作实践,为每个实践提供详细的方案描述和操作指导,帮助用户轻松构建基于CAE 的应用托管业务。

表 1-1 CAE 最佳实践一览表

| 最佳实践 | 说明 |
|--|--|
| 使用CAE托管Nginx静 态文件服务器 | 本章节介绍如何使用CAE托管Nginx静态文件服务器。您 可以将业务代码部分制作镜像后部署到CAE,静态文件部 分存储到与该组件关联的并行文件系统中,即可实现混合 业务和静态文件的前端组件托管。 |
| Gitlab对接Jenkins自动 构建并部署到CAE | 本章节以Java项目的构建、部署为例,介绍如何完成"完 整的代码提交 > Jenkins构建 > 软件包上传/镜像上传 > CAE部署"流程。 |
| Jenkins流水线支持多组 件按照依赖顺序部署 | 本章节介绍如何使用jenkins来构建部署升级微服务组 件,升级涉及多个微服务组件。 |
| SpringCloud应用部署 到CAE自动对接Nacos 引擎 | 本章节通过CAE源码部署能力自动化部署一个provider服 务和一个consumer服务,帮助您体验自动接入Nacos引 擎。 |
| Spring Cloud应用无损 上线 | 在组件运维过程中,不可避免要进行升级、重启、扩容等 操作,在这些操作中,无损上线是常见的要求。本章节介 绍如何配置Spring Cloud无损上线。 |
| 健康检查 | 本章节介绍如何通过CAE提供三种健康检查机制(存活探 针、就绪探针和启动探针)来检测您的应用实例是否正常 工作,保障业务正常运行。 |
| 生命周期管理 | 生命周期管理是用于在特定阶段执行调用的方法。本章节 介绍如何通过CAE提供的两种生命周期管理(启动后处 理、停止前处理)来处理事件。 |
| 发送事件告警到企业微 信 | 本章节介绍如何通过设置事件通知规则,可以帮助您及时 了解组件运行时的状态,快速定位问题。 |

| 最佳实践 | 说明 |
|--------------------------------------|--|
| 对接软件开发生产线 CodeArts流水线自动升 级到CAE | CAE目前提供了Codearts商业插件,可对接Codearts流水 线自动升级组件到CAE。本章节将根据不同场景为您介绍 "CAE升级插件"的使用方法。 |
| 通过配置PromQL实现 自定义弹性伸缩 | 假设某个组件提供自定义指标 http_requests_total(表示 http请求总量),本章节以该指标为例,介绍如何使用 PromQL实现自定义弹性伸缩。 |
| 配置CAE对接DEW,帮 助应用从DEW获取加密 凭据 | 本章节介绍如何配置CAE对接DEW,帮助应用从DEW获 取加密凭据。 |
| ASP.NET Core应用部署 到CAE | CAE源码部署支持Docker运行时,您可以自行配置 Dockerfile文件,在Dockerfile中安装构建环境,定义构 建命令,以此支持更多编程语言的项目在CAE部署。 本章节以以ASP.NET Core应用为例,介绍如何将ASP.NET Core应用部署到CAE。 |
| 通过企业路由打通网络 | CAE提供CAE环境访问租户侧VPC的能力,本章节将指导 您在关闭公网后,如何通过企业路由打通网络。 |

2 使用 CAE 托管 Nginx 静态文件服务器

2.1 概述

应用场景

本服务适用于业务代码及静态页面混合WEB前端组件托管场景。用户可以将业务代码 部分制作镜像后部署到CAE,静态文件部分存储到与该组件关联的并行文件系统中, 即可实现混合业务和静态文件的前端组件托管。组件部署后,用户可以通过更新并行 文件系统中的静态页面文件来实现实时更新前台应用。

此方案使用的Nginx版本为 "alpine-perl",此版本已在开源镜像提供。

图 2-1 镜像版本

| NGIN | 🕻 nginx 🗇 | | | | |
|------------|-------------|-------------------------------|---------|-----|---------------|
| 描述 | (| Official build of Nginx. | | | 1,254,285,934 |
| 更新时间 | : | 2018/11/07 21:47:28 GMT+08:00 | | 收藏量 | 11,785 |
| 描述 | 镜像版本 | | | | |
| | | | | | |
| 镜像版本 | | | 大小 | | |
| alpine-per | ł | | 15.7 MB | | |
| 1-alpine-p | erl | | 15.7 MB | | |
| mainline-a | alpine-perl | | 15.7 MB | | |
| alpine | | | 7.4 MB | | |
| | | | | | |

方案架构

Nginx是一个轻量级的web服务器,本身也是一个静态资源的http服务器。本实践以 Nginx为例,通过配置云存储中的并行文件系统来实现静态文件的托管,并通过更新并 行文件系统中的静态文件来实时更新Nginx访问页面。

图 2-2 Nginx 访问关系图



Nginx 默认配置

查询nginx默认配置方法:

Nginx默认配置如图2-3所示。

此版本的Nginx配置文件(nginx.conf)地址为"/etc/nginx/nginx.conf"。

图 2-3 nginx.conf

```
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log warn;
pid
             /var/run/nginx.pid;
events {
    worker_connections 1024;
http {
     include
                     /etc/nginx/mime.types;
     default_type application/octet-stream;
    log_format main '$remote_addr - $remote_user [$time_local] "$request"
'$status $body_bytes_sent "$http_referer" '
'"$http_user_agent" "$http_x_forwarded_for"';
     access_log /var/log/nginx/access.log main;
     sendfile
                        on;
     #tcp_nopush
                        on;
     keepalive_timeout 65;
     #gzip on;
     include /etc/nginx/conf.d/*.conf;
```

Nginx默认的default.conf如<mark>图2-4</mark>所示。"/usr/share/nginx/html"为静态文件存放的 路径,该路径下存在两个默认文件分别为"50x.html"和"index.html"。

图 2-4 default.conf

| server { | | |
|------------------------|--------------------------------|-------|
| listen | 80; | |
| server_name | localhost; | |
| | | |
| #charset koi | 8-r; | |
| <pre>#access_log</pre> | /var/log/nginx/host.access.log | main; |
| | | |
| location / { | | |
| root / | usr/share/nginx/html; | |
| index i | ndex.html index.htm; | |
| } | | |
| 2 | | |

如需替换页面显示,则将需替换的html文件放入"/usr/share/nginx/html"目录下。 替换的html文件可以通过<mark>获取静态文件</mark>提供的链接中直接下载。

2.2 部署前准备

创建环境

- 步骤1 登录CAE控制台。
- 步骤2 选择以下任意方式创建环境。
 - 在您首次使用本服务时,页面会提醒您尚未创建环境。
 - a. 单击**创建环境**卡片中的"立即创建"。

图 2-5 创建环境

| (1) 元应用引擎 | | | |
|---|--|---|---------|
| 云应用引擎 CAE 云应用引擎是一个面向应用的Serverless托管服务,提供极速部署 性伸幅、按量行费。可做到基础设施免运维,根据可观测的运行排 | 、极低成本、极简运输的一站式应用托雷方案。 | 源码、软件包、请像包块遮发布应用,砂圾弹 | - |
| 云应用引擎支持客户通 背 | 您还没有创建环境 這些环境命名,来自动新疆一个仅供当前相户使用的基础 高,计算,存储创厚,在此环境中的服务共用这个环境下() | 設課題,包括:提供应用运行的网 的基础资源。 | |
| (1) 创建环境 环境和于预高您创课的组件、通过给环 境命名来自动新建一个仅供当前租户使 用的基础资源组。 <u>立即创建</u> | (3) 第個課項用,一个应用總修 識的业务服务,可包含多个 經常能、支存儲得 | 4 部署组件 在当前应用和环境中部 包或者镜像包。 | 署代码源、软件 |

b. 在弹出的对话框中输入对应的参数,具体参照表2-1。

表 2-1 创建环境

| 参数 | 说明 |
|-------|--|
| 环境名称 | 输入自定义的环境名称。 |
| 企业项目 | 设置企业项目。 企业项目管理提供了一种按企业项目管理云资源的方 式,帮助您实现以企业项目为基本单元的资源及人员 的统一管理,默认项目为default。 <mark>开通企业项目</mark> 后可以使用。 |
| 虚拟私有云 | 下拉框中选择环境资源所在VPC。 如需创建VPC,请单击"创建虚拟私有云",具体操作 参考 <mark>创建虚拟私有云</mark> 。 注意: 环境创建完成后,不支持修改VPC。 |

| 参数 | 说明 |
|-----|--|
| 子网 | 下拉框中选择环境子网。 |
| | 无可用子网时,单击"创建子网",进入网络控制台 创建新子网,具体操作参考 <mark>为虚拟私有云创建新的子</mark> <mark>网</mark> 。 |
| | 注意: 子网需要保留至少2个可用网络IP地址,以供 CAE配置和优化使用,如果不满足条件,会创建失败。 |
| 安全组 | 选择"自动生成"。 |
| | 说明 安全组需要放通所选择的子网到子网网关地址,以及需要访 问的中间件如RDS,CSE等服务的访问地址和端口。 |
| 组织 | 如果您是首次使用本服务,在该下拉框中单击"创建 组织",输入自定义的组织名称。 |

- 非首次使用本服务,选择"组件列表"。
 - a. 单击页面上方环境模块右侧 🕀 。
 - b. 在弹出的"新增环境"对话框中输入新增环境的名称。

```
🛄 说明
```

可直接使用已创建的环境。

步骤3单击"确定",完成创建环境。

----结束

创建应用

步骤1 在导航栏中选择"组件列表",选择以下方式创建应用。

- 单击页面上方应用模块右侧 🕀 。
- 单击"组件列表"页面中的"立即创建"。

图 2-6 创建应用

| 云应用引擎 | 🚺 环境: 🕤 new | ▼ ⊕ | • (+) |
|-------|-------------|-----|----------------------------|
| 概览 | | | |
| 组件列表 | | | |
| 实例列表 | | | |
| 组件配置 | | | _ |
| 组件事件 | | | ! |
| 组件监控 | | | - Q |
| 组件日志 | | | 当前没有应用 您需要先创建一个应用, 立即创建 |
| 系统设置 | 4 | | |
| | | | |

步骤2 在弹出的"新增应用"对话框中输入新增应用的名称。

步骤3 单击"确定",完成创建应用。

----结束

获取静态文件

您可以通过以下链接下载"index.html"和"test.html"静态文件进行更新操作。

下载地址: https://cae-demo.obs.cn-north-4.myhuaweicloud.com/nginx.zip或 https://cae-demo.obs.cn-north-4.myhuaweicloud.cn/nginx.zip

2.3 操作步骤

约束与限制

官方Nginx镜像使用nginx用户运行,您需要将挂载文件的文件掩码(umask)设置为 0022,确保nginx用户拥有读取挂载文件的权限,指导操作请参考<mark>配置云存储挂载路</mark> 径。

创建并部署 Nginx 组件

- 步骤1 登录CAE控制台。
- 步骤2 选择"组件列表"
- 步骤3 在页面上方,下拉选择已创建的应用和环境,单击"新增组件"。
- 步骤4 参考表2-2设置组件信息。

表 2-2 组件基本信息

| 参数 | 说明 |
|------|---|
| 组件名称 | 新建组件的名称。本实践输入名称为"nginx"。 |
| 版本号 | 组件的版本号。 本实践版本号为1.0.0。 |
| 实例规格 | 选择实例规格,例如:0.5core、1GiB。 |
| 实例数量 | 输入实例数为1。 |
| 代码源 | 选择"镜像 > 开源镜像 > nginx"。此方案使用的nginx的版本为 "alpine-perl"。 |

图 2-7 创建组件

创建组件

| 组件配额为 50 个, | 您还可以 | 创建 47 个组 | 3件; 如壽申请更多 | 和额清点击中被 | 扩大配 | Ő | | |
|-------------|------------|--|------------------|---------|-----|-------------------|---------------|-----|
| 组件名称 | nginx | | | | | | | |
| 版本号 | 1.0.0 | | | | | | | |
| 实例规格 | 0.5 Co | re | • | 1 GiB | | • | | |
| 突例数量 | _ | 1 + |] | | | | | |
| 代码源 | 194 194 | 明仓库 | 鏡像 | 软件包 | の前 | 主容器镜像服务SWF | R控制台进行管理 | |
| | 我的镜 | 象 开》 | 原镜像 共調 | 享镜像 | | | | |
| | | | | | | | 镜像名称搜索 | Q C |
| | | ✓ ^{ubunt} 镜像 | u | | | | | |
| | | へ ・ ・ ・ ・ ・ ・ の ginx | | | | | | |
| | | | 版本 | | | 上传时间 🧜 | | 操作 |
| | | ۲ | alpine-perl | | | 2018/11/07 21:45: | :10 GMT+08:00 | 删除 |
| | | 0 | 1-alpine-perl | | | 2018/11/07 21:45: | :09 GMT+08:00 | 删除 |
| | | 0 | mainline-alpine- | perl | | 2018/11/07 21:45: | :09 GMT+08:00 | 删除 |
| | | 0 | alpine | | | 2018/11/07 21:44: | 34 GMT+08:00 | 删除 |
| | | 0 | 1-alpine | | | 2018/11/07 21:44: | 33 GMT+08:00 | 删除 |
| | | 5 * | 总条数:5 〈 | (1) | | | | |

● 健康检查:使用默认值"启动"。

- 负载均衡器:选择"内置负载均衡器"。
- 步骤7 在"负载均衡配置"页签下,单击"添加负载均衡配置"并设置参数。

步骤6 在"组件配置"页面,单击"访问方式"模块的"编辑",对Nginx组件进行配置。

版权所有 © 华为云计算技术有限公司

步骤5 单击"配置组件"。

文档版本 01 (2025-01-22)

- 协议:选择"TCP"。
- 访问控制:使用默认值"允许所有IP访问"。

9

- 访问端口: 自定义访问端口。本实践设置为"14632"。
- 监听端口:输入"80"。 •

图 2-8 添加负载均衡外网访问

| 负载均衡器 | 内置负载均衡器 ▼ | | | |
|-------|------------------------|-------------------------|--------|--------|
| 健康检查 | 不启用 | | | |
| | 协议: TCP 检查周期 (秒) : 5 | 超时时间 (秒) : 10 最大重试为 | 成数:3 🙋 | |
| 访问控制 | 允许所有IP访问 | • | | |
| 端口配置 | 145 MVr | 14- 87 ⁻ 240 | | +12 // |
| | MNX | 溢 WT 3% LI | 加加別所口 | 採作 |
| | TCP 🔻 | 80 | 14632 | 删除 |
| | (+) 添加端口配置 | | | |

步骤8 单击"确定",完成对该组件访问方式的配置。

步骤9 在"组件配置"页面,单击"配置并部署组件",使Nginx组件访问方式生效。

步骤10 部署成功后,在左侧导航栏中选择"组件列表",进入"组件列表"页面。

单击对应组件"访问地址"列的ip地址,查看Nginx静态WEB页面。

图 2-9 访问静态页面

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to <u>nginx.org</u>. Commercial support is available at <u>nginx.com</u>.

Thank you for using nginx.

----结束

云存储授权

步骤1 选择"系统设置 > 云存储授权",单击"编辑",打开"已授权对象存储"弹框。

步骤2 单击"授权并行文件系统",进入"授权并行文件系统"页面。

图 2-10 授权并行文件系统

已授权云存储

| 1 温馨提示:以下云存储料 | \$供给各组件使用,但需在 | 组件配置中完成配置。 | | | | × |
|-------------------|---------------|-------------------------------|------------------------|---------|----|----|
| ① 授权并行文件系统 | ① 授权存储桶 | 受授权通用文件系统 | 0 | 请输入关键词 | | QC |
| 名称 | 类型 | 容量 | 导入时间 🖕 | | 操作 | |
| test-sfs-othervpc | 通用文件系统 | | 2024/03/06 19:14:34 GM | T+08:00 | 解绑 | |
| test-0201 | 并行文件系统 | 0.00MB | 2024/03/01 19:22:39 GM | T+08:00 | 解绑 | |

- 步骤3 单击"创建并行文件系统",在输入框中输入自定义名称,例如: test-nginx。
- 步骤4 单击"确认"。
- 步骤5 选择已创建的并行文件系统。您还可以通过列表上方搜索框搜索关键词筛选数据。
- 步骤6单击"授权"。

在访问密钥授权弹框,输入访问密钥ID(AK)和秘密访问密钥(SK),并单击"确 认"。

可单击"获取访问密钥",获取访问密钥AK/SK,具体操作请参考访问密钥。

🛄 说明

如果您的环境是2024年4月20日前创建的,直接跳过此步骤。

步骤7 当页面显示"授权成功",可在"已授权云存储"页面查看授权成功的并行文件系统。

----结束

上传文件

- 步骤1 登录OBS控制台。
- **步骤2** 在左侧导航栏选择"并行文件系统",单击名称为"test-nginx"的文件系统,进入该并行系统页面。

图 2-11 并行文件系统

| 对象存储服务 | | 并行文件系统 ⑦ 开闢 | 声明 | | | |
|-----------|---|---------------|-------------|---------------|----------------------|-------------|
| 总资 NEW | | | | | | |
| 桶列表 | | 忽的帐号及帐号下的所有 | IAM用户可创建的稀释 | 如开行又件系统的上限为10 | 10个。建议您使用确内又件买管理项目区分 | ,合埋进行補规划和使用 |
| 并行文件系统 | | 您还可以创建0个并行文件新 | 经先。 | | | |
| 总用 NEW | | 通过指定属性的关键字搜 | 劳. 武 | | | |
| 资源包管理 NEW | | 文件系统名称 ⑦ ↓Ξ | 特色功能 | 叉 区域 | ▽ 数据冗余存储策略 | 存储用量 ⑦ 1三 |
| 我的蹇犧 | ø | test-nginx | | | 单AZ存储 | 187.45 KB |
| 数据快递服务 | æ | tesbooodtt | | | 单AZ存储 | 641 bytes |
| 云存储网关 | o | test-hjm | 2 | | 单AZ存储 | 2.51 KB |

步骤3 在左侧导航栏选择"文件",单击"上传文件"。

- **步骤4** 将已获取的静态文件"index.html"、"test.html"拖拽至文件上传处,也可单击 "添加文件"上传。存储类别默认为"标准存储"。
 - **图 2-12** 上传文件

| < test-nginx 🗇 | | | |
|------------------|---|---|---|
| 概点 | 文件 🗇 | | × |
| 文件 | | 上传文件 超过5GB如何上传? | |
| | 文件 碎片 | Lip文件 ——— ② 識額量 (可識) | |
| 访问权限控制 | 若需要培文件移动到并行文件系统内其他位置, 基于名 <mark>1</mark> 合初要求,从浏览器吉接访问文件时不 | 1 上標語作時产生演奏费用,上標或功能將产生存結费用, | × |
| 基础配置 ▼ | 上传文件 新建文件夹 恢复 | ▲ 当前并行文件系统开启了访问策略。如果访问策略允许公共读或公共读写,上传的文件可能存在数据安全风险。 | × |
| | 名称 | 存施关制 标准存储 | |
| | | 2 | |
| | | 上使文件 ●注意:并行文件系统对如有同名文件, 裕振新上统的文件 覆盖 。 | |
| | | OBS | |
| | | | |
| | | | |
| | | 指摘本地文件框支件类型此处,或添加文件 (単次最多支持100个文件调时上得,总大小不服过5GB) | |
| | | | |
| | | 下一步: 高級計畫 (阿逸) | Ä |

步骤5 单击"上传",完成文件上传。

----结束

配置云存储挂载路径

步骤1 返回CAE控制台,选择"组件配置"。

步骤2 在"组件配置"页面上方的下拉框中选择Nginx组件。

步骤3 单击"云存储配置"模块中的"编辑",进入云存储配置页面。

图 2-13 配置云存储

| 前 云应用引擎 | ☑ 环境: ● nacre-env1 ▼ 🕘 📦 应用: app- | cxm 🔹 🕀 | 云存储配置 | |
|------------------|--|------------------------|-------------------------------------|---|
| 概选 组(4列表 | ● 提件: nginx / v1.0.0 ▼ | 600 | ① 配置并行文件 名称 | (1) (1) (1) (1) (1) (1) (1) (1) (1) (1) |
| 44年配置 组件事件 | 中间件 | | | |
| 组件监控 | | 微服务引擎CSE ⑦ | | |
| 组件口志 系统设置 | 202大学に大学が生まれ、「日本では、」 | 「四土木坊た」しつと、メ川線(以上加賀坊た。 | | |
| | 配置 | 配置 | | |

步骤4 在"云存储配置"页面单击"配置并行文件系统",并输入挂载路径并设置权限。

- 并行文件系统名称:选择云存储授权中授权的并行文件系统"test-nginx"。
- 文件掩码(umask): 文件掩码设置为0022。
- 挂载路径:数据存储挂载到组件上的路径。本实践使用"nginx"默认路径"/usr/share/nginx/html"。

权限: 挂载路径及挂载路径下文件, 有"读写"、"只读"两种权限, 此处选择 "读写"。

注意事项:

- 挂载静态文件路径到云存储上时需要注意:请不要挂载到含有系统文件的目录 • 下,如"/"、"/var/run"等,否则可能会导致部署的组件异常。
- 挂载路径选择权限时需要注意:读写权限表示组件对于该挂载路径及路径下的所 有文件拥有读写权限,只读即只有只读权限。

图 2-14 配置云存储挂载路径

| 配置并行文件系统 | | | | |
|-----------------------|-----------------------------|-----------------------------|------|----|
| 并行文件系统名称 | test-nginx 当前只支持选择类型为标准存 | ▼ C 去授权并行文件系统 储类型的并行文件系统 | | |
| 文件掩码 (umask) | 0022 | | | |
| 挂载路径 ? | | 子路径 ⑦ | 权限 | 操作 |
| /usr/share/nginx/html | | 请输入子路径,如:tmp | 读写 ▼ | 删除 |
| ⊕ 添加挂载路径 | | | | |

- 步骤5 单击"确定"后在"云存储配置"页面再次单击"确定",完成配置。
- 步骤6 在"组件配置"页面单击"生效配置"。

| 图 2-15 | 生效配置 |
|--------|------|
|--------|------|

| 现 范 目件版本 | ● 组件: nginx / v1.0.0 ▼ | OOC |
|--------------------|--|-------------------|
| 目件配置 | 中间件 | |
| 组件事件 | | |
| 组件监控 | 云数据库RDS ⑦ | 微服务引擎CSE ⑦ |
| 组件日志 | 您还未绑定云数据库RDS,如需使用RDS请绑定。 | 您还未绑定CSE,如需使用请绑定。 |
| 系统设置 | | |
| | | |
| | 配置 | 配置 |

| | 2二6社-000108 | |
|----|-------------|--|
| 结束 | | |
| | | |
| | | |

查看更新页面

文档版本 01 (2025-01-22)

○○ 说明

(单击F5刷新页面)。

- 步骤2 单击对应Nginx组件"访问地址"列的ip地址,再次访问Nginx,可以实时更新页面
- 步骤1 在左侧导航栏中选择"组件列表",进入"组件列表"页面。

版权所有 © 华为云计算技术有限公司

新的"index.html"文件可以根据您的需求自定义并上传。

图 2-16 更新后静态页面

Hello, Welcome to cae!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to <u>nginx.org</u>. Commercial support is available at <u>nginx.com</u>.

Thank you for using nginx.

您还可以通过/test.html访问新的静态页面文件。

图 2-17 访问 test 页面

100. :14632/test.html

Welcome to test!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to <u>nginx.org</u>. Commercial support is available at <u>nginx.com</u>.

Thank you for using nginx.

----结束

3 Gitlab 对接 Jenkins 自动构建并部署到 CAE

3.1 概述

背景

完成代码开发后,每次上线前需先在Jenkins上打包成镜像或软件包,再将镜像手动上 传到swr或者将软件包手动上传到obs,然后去CAE升级组件。该流程较为繁琐,频繁 发版测试导致开发和运维效率低,体验差,提供如下最佳实践提高开发效率。下面以 Java项目的构建、部署为例提供了完整的代码提交>>Jenkins构建>>软件包上传/镜像 上传>>CAE部署的指导,如涉及其他语言如Golang等只需对应安装编译构建环境即 可。

适用场景

此功能适用于您的代码已在Gitlab上管理,通过Jenkins构建打包,使用CAE进行应用托 管, 并且已经部署了组件之后,需要进行组件升级的场景。

第一次部署请参考创建组件。

解决方案

提供上传jar包和镜像并升级CAE组件的方案,输出一个shell脚本在Jenkins构建打包完成之后调用脚本自动部署到CAE环境中,实现代码合入后自动构建打包部署。

流程优化对比

当前上线流程图如下:



3.2 环境准备

3.2.1 准备 Jenkins 环境

环境信息说明

🛄 说明

如果已安装好jenkins环境,请跳过本章节。

在linux虚拟机上安装好Jenkins,本实践使用的具体环境信息如下所示。如果使用镜像 部署,需要在虚拟机中安装docker。

- 虚拟机: Centos7.9
- Jenkins: 2.331
- git: yum安装
- jdk: 11.0.8
- Apache Maven : 3.8.6

部署时需注意:

部署的Jenkins启动时需添加如下参数: -Dhudson.security.csrf.GlobalCrumbIssuerConfiguration.DISABLE_CSRF_PROTECTION=true

否则Gitlab对接Jenkins会失败,报错信息如下:

HTTP Status 403 - No valid crumb was included in the request

相关软件下载及安装

- Jenkins下载链接: https://mirrors.jenkins.io/war-stable/
- 安装git用于拉取代码进行构建: yum install git -y
- jdk安装包下载链接: https://www.oracle.com/cn/java/technologies/downloads/#java11
- maven安装包下载链接:
 https://maven.apache.org/download.cgi
- docker安装用于打包镜像并上传到镜像仓库: yum install docker

安装后检查

- 检查git:
 - 检查git: [root@ecs-jenkins ~]# **git version** git version 1.8.3.1
 - 检查JDK: [root@ecs-jenkins jar]# java -version openjdk version "1.8.0_345" OpenJDK Runtime Environment (build 1.8.0_345-b01) OpenJDK 64-Bit Server VM (build 25.345-b01, mixed mode)
 - 检查Maven: [root@ecs-jenkins jar]# **mvn -v** Apache Maven 3.8.6 (84538c9988a25aec085021c365c560670ad80f63) Maven home: /root/app/maven/apache-maven-3.8.6

Java version: 11.0.8, vendor: Huawei Technologies Co., LTD, runtime: /root/app/jdk11/jdk-11.0.8 Default locale: en_US, platform encoding: UTF-8 OS name: "linux", version: "3.10.0-1160.76.1.el7.x86_64", arch: "amd64", family: "unix"

- 检查Docker:

[root@ecs-jenkins jar]# docker version Client: Version: 1.13.1 API version: 1.26 Package version: docker-1.13.1-209.git7d71120.el7.centos.x86_64 go1.10.3 Go version: Git commit: 7d71120/1.13.1 Wed Mar 2 15:25:43 2022 Built: OS/Arch: linux/amd64 Server: Version: 1.13.1 API version: 1.26 (minimum version 1.12) Package version: docker-1.13.1-209.git7d71120.el7.centos.x86_64 Go version: go1.10.3 7d71120/1.13.1 Git commit: Wed Mar 2 15:25:43 2022 Built: OS/Arch: linux/amd64 Experimental: false

图 3-1 版本查验

| [root@ecs-jenkins | ~]# git version | | | |
|----------------------------|---|--|--|--|
| git version 1.8.3. | | | | |
| [root@ecs-jenkins | ~]# java -version | | | |
| openjdk version "1 | .8.0 345" | | | |
| OpenJDK Runtime En | vironment (build 1.8.0 345-b01) | | | |
| OpenJDK 64-Bit Ser | ver VM (build 25.345-b01, mixed mode) | | | |
| [root@ecs-jenkins | ~]# mvn -v | | | |
| Apache Maven 3.8.6 | (84538c9988a25aec085021c365c560670ad80f63) | | | |
| Maven home: /root/ | app/maven/apache-maven-3.8.6 | | | |
| Java version: 11.0 | .8, vendor: Huawei Technologies Co., LTD, runtime: /root/app/jdk11/jdk-11.0.8 | | | |
| Default locale: en | US, platform encoding: UTF-8 | | | |
| OS name: "linux", | version: "3.10.0-1160.76.1.el7.x86 64", arch: "amd64", family: "unix" | | | |
| [root@ecs-jenkins | ~]# docker version | | | |
| Client: | | | | |
| Version: | 1.13.1 | | | |
| API version: | 1.26 | | | |
| Package version: | docker-1.13.1-209.git7d71120.el7.centos.x86 64 | | | |
| Go version: | qo1.10.3 | | | |
| Git commit: | 7d71120/1.13.1 | | | |
| Built: | Wed Mar 2 15:25:43 2022 | | | |
| 0S/Arch: | linux/amd64 | | | |
| | | | | |
| Server: | | | | |
| Version: | 1.13.1 | | | |
| API version: | 1.26 (minimum version 1.12) | | | |
| Package version: | docker-1.13.1-209.git7d71120.el7.centos.x86_64 | | | |
| Go version: | go1.10.3 | | | |
| Git commit: 7d71120/1.13.1 | | | | |
| Built: | Wed Mar 2 15:25:43 2022 | | | |
| 0S/Arch: | linux/amd64 | | | |
| Experimental: | false | | | |
| [root@ecs-jenkins | ~]# | | | |

3.2.2 上传代码到 Gitlab 代码仓库

本实践使用的是Java项目代码,使用Maven构建Jar包。

前提条件

- 1. Jenkins所在Linux虚拟机能够访问GitLab代码仓库。
- 2. 已经在GitLab创建账号和仓库。

操作步骤

步骤1 登录GitLab。

文档版本 01 (2025-01-22)

步骤2 上传代码到已创建好的代码仓库。

----结束

3.2.3 安装 obsutil 工具

🛄 说明

如果不涉及软件包部署,请跳过本章节。

前提条件

- 获取AK/SK,请参考AK/SK获取方法。
- 已获取部署组件的CAE所在区域的终端节点,参考地区和终端节点。
- 已在和部署组件的CAE在同一区域的OBS中创建桶,用于存储软件包,具体操作请参见创建桶。示例选择的桶名为cae-obs。

下载和安装

- 操作系统的选择:
 下载安装前在Jenkins所在虚拟机中执行命令查看虚拟机操作系统类型:
 echo \$HOSTTYPE
 - 若执行如上命令的输出值是"x86_64",请下载AMD 64位系统;
 - 若执行如上命令的输出值是"aarch64",请下载ARM 64位系统。
 - obsutil工具用于构建后上传软件包到obs,请参见下载和安装obsutil。

初始化配置

执行以下配置命令,初始化配置obsutil工具。 {path}/obsutil config -i=ak -k=sk -e={endpoint}

其中:

- {path}需要替换为obsutil安装路径,例如:/root/tools/obsutil/ obsutil_linux_amd64_5.4.6。
- {endpoint}需要替换为已获取到的部署组件的CAE所在区域的终端节点。

验证 obsutil 上传文件到 OBS 是否正常

- **步骤1** 创建测试文件。例如:test.txt。 touch test.txt
- 步骤2 使用obsutil上传创建的文件到obs。

~/tools/obsutil/obsutil_linux_amd64_5.4.6/obsutil cp test.txt obs://{OBS桶名称}

请将{OBS桶名称}替换为已创建的待使用的OBS桶名称,本示例选择的桶名为caeobs,将在当前目录新建的test.txt文件上传到cae-obs桶中。提示"Upload successfully"表示上传成功。

图 3-2 上传文件到 obs

| [root@ecs-jenk: | ins test]# ~/tools/ob | bsutil/obsutil | linux amd64 5.4.6/obsutil cp test.txt obs://cae-obs |
|-----------------|-----------------------|-----------------|--|
| Start at 2022- | 10-12 08:42:28.23075 | 7928 +0000 UTC | |
| | | | |
| | | | |
| Parallel: | 5 | Jobs: | 5 |
| Threshold: | 50.00MB | PartSize: | auto |
| VerifvLenath: | | VerifvMd5: | |
| CheckpointDir: | /root/.obsutil check | kpoint | |
| | | | |
| | | |] 100.00% ?/s 12B/12B 114ms |
| | | | |
| Upload success | fully, 12B, n/a, /roo | ot/test/test.tx | c> obs://cae-obs/test.txt, cost [113], status [200], request id [00000183CB5C020F4011AE3721CFAFE5] |
| [root@ecs-jenk: | ins test]# | | |

- 步骤3 登录OBS控制台,选择"桶列表"。
- 步骤4 单击本示例桶名名称cae-obs,进入"对象"页面,查看已经上传的文件test.txt。

3.2.4 安装 KooCLI 工具

KooCLI工具用于调用CAE服务提供的接口,对CAE组件执行升级等操作。

使用KooCLI工具之前,您需要先安装和初始化配置KooCLI工具:

- 安装KooCLI:您可以选择**方式一:联网安装**或者<mark>方式二:软件包安装</mark>安装KooCLI 工具。
- 初始化配置KooCLI:使用KooCLI工具前,需要先进行初始化配置。

方式一: 联网安装

步骤1 登录Jenkins所在虚拟机.

步骤2 执行安装命令:

curl -sSL https://hwcloudcli.obs.cn-north-1.myhuaweicloud.com/cli/latest/hcloud_install.sh -o ./ hcloud_install.sh && bash ./hcloud_install.sh -y

🛄 说明

如上命令默认将KooCLI下载至"/usr/local/hcloud/"目录下,同时在"/usr/local/bin/"目录下 创建KooCLI的符号链接。

----结束

方式二: 软件包安装

步骤1 登录Jenkins所在虚拟机,执行如下命令确认所需安装操作系统:

echo \$HOSTTYPE

- 若执行如上命令的输出值是"x86_64",请下载AMD 64位系统;
- 若执行如上命令的输出值是"aarch64",请下载ARM 64位系统。
- 步骤2 执行如下命令下载对应的软件包。
 - AMD

wget "https://hwcloudcli.obs.cn-north-1.myhuaweicloud.com/cli/latest/huaweicloud-cli-linux-amd64.tar.gz" -O huaweicloud-cli-linux-amd64.tar.gz

• ARM

wget "https://hwcloudcli.obs.cn-north-1.myhuaweicloud.com/cli/latest/huaweicloud-cli-linux-arm64.tar.gz" -O huaweicloud-cli-linux-arm64.tar.gz

步骤3 执行如下命令解压软件包。

• AMD

tar -zxvf huaweicloud-cli-linux-amd64.tar.gz

- ARM tar -zxvf huaweicloud-cli-linux-arm64.tar.gz
- 步骤4 在解压后的目录执行如下命令创建软链接到"/usr/local/bin"目录: In -s \$(pwd)/hcloud /usr/local/bin/
- 步骤5 执行如下命令验证是否安装成功:

hcloud version

系统显示类似"当前KooCLI版本:3.4.1.1"版本信息,表示安装成功。

图 3-3 koocli 版本查验

[root@ecs-jenkins ~]# hcloud version 当前KooCLI版本:3.4.1.1

----结束

验证安装结果

步骤1 KooCLI工具安装完成后,请执行hcloud CAE命令测试是否支持CAE服务。

LrootBhecs- J "JH hcloud CAE LUSE_ERRORJUnsupported service: CAE. Run 'hcloud --help' for the KooCLI operation guide. Run 'hcloud --interactive' for the interactive prompt for building comma interactive communications of the second second

- **步骤2** 如果出现**Unsupported service: CAE**,则需要将KooCLI工具切换成中文。 hcloud configure set --cli-lang=cn
- 步骤3 切换完成后,再次执行hcloud CAE命令。
- 步骤4 如果没有出现ERROR,则表示KooCLI工具已支持CAE服务。

----结束

初始化配置 KooCLI

- 步骤1 登录Jenkins所在Linux虚拟机。
- **步骤2**执行命令进行初始化配置,输入命令后按回车进入交互模式,根据界面提示输入各参数值,各参数配置参考表3-1。 hcloud configure init

表 3-1 初始化配置

| 参数 | 说明 |
|-------------------|--|
| Access Key ID | (必填参数)访问密钥ID,即AK。获取方法,请参考 <mark>访</mark> <mark>问密钥</mark> 。 |
| Secret Access Key | (必填参数)与访问密钥ID(AK)结合使用的密钥,即 SK,初始化时必填。获取方法,请参考 <mark>访问密钥</mark> 。 |
| Region | (选填参数)区域,即CAE服务部署区域。获取方法, 请参考 <mark>地区和终端节点</mark> 。 |

----结束

3.2.5 安装插件并配置 Jenkins 工具

在使用GitLab使用Gitlab对接Jenkins自动构建并部署组件到CAE前,需要安装Jenkins 插件和并配置Jenkins全局参数。

- **安装Jenkins插件**:用于对接git以及支持在构建的时候使用脚本。
- Jenkins全局工具配置:用于Jekins流水线打包脚本对接git拉取代码并打包。

安装 Jenkins 插件

- 步骤1 在浏览器地址栏输入http://{安装Jekins的Linux虚拟机IP}:8080,登录Jenkins。
- 步骤2 选择"系统管理 > 插件管理"。
- **步骤3**单击"可选插件",搜索表3-2中的插件进行安装。

表 3-2 插件安装说明

| 插件名称 | 是否必须 | 说明 |
|---------------------------------------|------|-------------------------|
| Generic Webhook Trigger Plugin | 是 | 用于对接Gitlab的 webhook |
| GitLab Plugin | 是 | 允许Gitlab触发Jenkins构 建 |
| Pipeline: Basic Steps | 是 | 支持pipeline脚本语法 |
| Pipeline: Build Step | 是 | 支持pipeline脚本语法 |
| Pipeline: Stage Step | 是 | 支持pipeline脚本语法 |
| Localization: Chinese (Simplified) | 否 | 简体中文语言包 |

图 3-4 Jenkins 系统管理页面



图 3-5 Jenkins 插件管理页面

| 🧛 Jenkins | |
|---|-----------------------------------|
| Dashboard , 插件管理 | |
| ▲ 返回工作台 ◆ 管理 Jenkins | Plugin Manager 可更新 可选插件 已安装 高级 |
| | Install Name 1 |
| <i></i> | |

----结束

Jenkins 全局工具配置

步骤1 选择"系统管理 > 全局工具配置"。

| 图 3-6 | 全局工具配置 |
|-------|--------|
|-------|--------|

| 🏘 Jenkins | | | Q 查找 | ⑦ ≜ admin 등 |
|---------------|--|---|------|--|
| Dashboard + | | | | |
| 新建任务 | 管理Jenkins | | | |
| ≥ 构建历史 | Building on the built-in node can be a security issue. You should | set up distributed builds. See the documentation. | | Set up agent Set up cloud Dismis |
| 🔍 项目关系 | | | | |
| ▲ 检查文件指纹 | 物形没有配置時始當求的面前戶,这可能会有安全问题。更多信息,请参考 <mark>这个页面</mark> 。 本の回該方法の目前時代書書 時間時代 またれが必然である。 | | | 不再显示 |
| 🎲 系统管理 | APPALX IN BUILDER APPALING | | | |
| | | | | |
| * 新建规图 | 系统配置 | | | |
| 构建队列 ^ | 系统配置 配置全局设置和路径 | 全局工具配置 工具配置,包括它们的位置和自动安装器 | ÷ | 插件管理 添加、删除、禁用或启用Jenkins功能扩展插件。 |
| 队列中设有构建任务 | | | | |
| 格建执行状态 ^ | 节点管理 添加、删除、控制和监视系统运行任务的节点。 | | | |

步骤2 配置maven。

示例中的maven安装目录"/root/app/maven/apache-maven-3.8.6",请获取您的实际Maven安装目录。

图 3-7 Maven 配置

Maven 配置

默认 settings 提供

文件系统中的 settings 文件

文件路径 ?

/root/app/maven/apache-maven-3.8.6/conf/settings.xml

默认全局 settings 提供

文件系统中的全局 settings 文件

文件路径 ?

/root/app/maven/apache-maven-3.8.6/conf/settings.xml

图 3-8 Maven 安装

| laven | |
|------------------------------------|--|
| aven 安装 | |
| 新增 Maven | |
| Maven | |
| Name | |
| maven | |
| MAVEN_HOME | |
| /root/app/maven/apache-maven-3.8.6 | |
| □ 自动安装 ? | |

步骤3 配置JDK。

示例中的JDK安装目录"/root/app/jdk11/jdk-11.0.8",请获取您的实际JDK安装目录。

图 3-9 JDK 安装

| JD | K |
|-----|----------------------------|
| JDI | く安装 |
| | 概論 JDK |
| | JDK |
| | 别名 |
| | [idk11 |
| | JAVA_HOME |
| | /root/app/jdk11/jdk-11.0.8 |
| | |
| | |

步骤4 配置Git。

示例中的Git工具目录"/usr/bin/git",请获取您的实际Git安装目录。

图 3-10 Git 安装

| Sit | | |
|-------|--------------------------|------------|
| iit i | installations | |
| | Git | |
| | Name | |
| | git | |
| | Path to Git executable 👔 | |
| | /usr/bin/git | |
| | □ 自动安装 ? | |
| | | Delete Git |

----结束

3.3 操作步骤

3.3.1 操作前对接测试

前提条件

操作前需进行Jenkins对接Gitlab测试,保证Jenkins通过API访问Gitlab没有问题。

生成 Gitlab 访问令牌

- 步骤1 登录Gitlab。
- 步骤2 鼠标移动到右上角的账号名上,单击"Edit profile"。

图 3-11 进入编辑页面



步骤3 单击 "Access Tokens",输入 "Token name",勾选 "api",单击 "Create personal access token" 创建访问令牌。

| 图 3-1 | 2 创 | 建访 | 问令牌 |
|-------|-----|----|-----|
|-------|-----|----|-----|

| User Settings | Open registration is enabled on your instant learn more about how you can customize / | tio. X disable registration on your instance. |
|--------------------|--|--|
| (8) Profile | 10-mu antilian | |
| 8* Account | view second | |
| 88 Applications | User Settings > Access Tokens | |
| Chat | | |
| Access Tokens | Q Search settings | |
| 🖸 Emails | Personal Access Tokens | Add a personal access token |
| Password | You can generate a gersonal arress token for | Enter the name of your application, and we'll return a unique personal access token. |
| △ Notifications | each application you use that needs access to | Token name |
| 🖉 SSH Keys | the GitLab API. | |
| P GPG Keys | You can also use personal access tokens to | For example, the amination using the token or the summer of the token |
| 82 Preferences | autorenotate against on over HTTP. They are the only accepted password when you have Two- | Exclusion data |
| C Active Sessions | Factor Authentication (2FA) enabled. | NOV NY CO |
| Authentication log | | 1111-Mill-OD |
| | | Select scopes |
| | Г | scopes set the permission levels granted to the toker, Leann more. |
| | | api Grants complete read/write access to the API, including all groups and projects, the container registry, and the package registry. |
| | L. | read_user Grant read-only access to the authenticated user's profile through the Juser API endpoint, which includes usersame, public email, and full name. Also grants access to read-only API endpoints under Juses. |
| | | read.api Gunts were access to the API, including all groups and projects, the container registry, and the package registry. |
| | | Grants and separationy Grants and only access to repositories on private projects using Git-over-HTTP or the Repository Res AP, |
| | | with separatory Grants read-write access to repositories on private projects using Git-over-HTTP (not using the APD. |
| | | Guido partición to perform API actions as any user in the system, when authenticated as an admin user. |
| | | Create personal access token |
| | | Active personal access tokens (1) |
| | | Personal access tokens are not revoked upon expiration. |

完成后在页面上方的"Personal Access Tokens"右侧显示token令牌。

注意: 令牌仅在初次生成时显示,否则下次需要重新创建。该令牌仅用于Gitlab对接测 试。

图 3-13 令牌显示

| User Settings > Access Tokens | | | | |
|---|---|---|--|--|
| ① Your new personal access token has been on | eated. | × | | |
| Q Search settings | | | | |
| Personal Access Tokens You can generate a personal access token for each application you use that needs access to the GitLab API. | Your new personal access token 4 Make sure you save it - you won't be able to access it again. | ß | | |
| You can also use personal access tokens to authenticate against Git over HTTP. They are the only accepted password when you have Two- Factor Authentication (2FA) enabled. | Add a personal access token Enter the name of your application, and we'll return a unique personal access token. Token name | | | |

----结束

Jenkins 对接 Gitlab 测试配置

步骤1 在浏览器地址栏输入http://{安装Jekins的Linux虚拟机IP}:8080,登录Jenkins。

步骤2 选择"系统管理 > 系统配置",在配置中选择"Gitlab"。

图 3-14 进入 Gitlab 配置页面

| 🙀 Jenkins | | | |
|-------------------|--|---------|--|
| Dashboard 🔸 j | 配置▼ | | |
| | Jenkins Location | | |
| 音 新建任务 | Serve resource files from another domain | £ ? | |
| & 用户列表 | 全局属性 | jenkins | |
| | Pipeline Speed/Durability Settings | | |
| 📂 构建历史 | 使用统计 | 息 ? | |
| | Folder | | |
| ₩ 坝日大糸 | Gitlab | | |
| 🏬 检查文件指纹 | Timestamper | | |
| · | Fingerprints | | |

步骤3 配置Gitlab的url,并单击Credentials下方的"添加",选择"Jenkins"。

图 3-15 配置 Gitlab URL

Enable authentication for '/project' end-point

GitLab connections

Connection name

gitlab

A name for the connection

Gitlab host URL

http://1

The complete URL to the Gitlab server (e.g. http://gitlab.mydomain.com)

Credentials



步骤4 在下拉框单击"Username with password",选择"Gitlab API token",将<mark>生成</mark> Gitlab访问令牌中Gitlab的访问令牌配置到API token中。

图 3-16 选择凭据类型

| 🕋 Jenkins 凭据提供者: Jenkins |
|---|
| ee Add Credentials |
| 全局凭据 (unrestricted) |
| Kind Username with password |
| Username with password |
| Gillab API token Jenkins Build Token SSH Username with private key Secret file Secret text Certificate |

图 3-17 配置访问令牌

| 🕋 Jenkins 凭据提供者: Jenkins | | | | |
|---|--|--|--|--|
| ed Add Credentials | | | | |
| 全局凭据 (unrestricted) | | | | |
| Kind | | | | |
| GitLab API token | | | | |
| Scope ? 全局 (Jenkins, nodes, items, all child items, etc) | | | | |
| API token | | | | |
| 9 GI | | | | |
| Description ? | | | | |
| Add Cancel | | | | |

步骤5 Credentials选择"Gitlab API token",单击"Test Connection",返回"Success" 表示成功。

----结束

3.3.2 配置流水线构建任务

- 场景一:使用Jenkins构建生成的是软件包,如Jar包,就使用脚本中的软件包部署 场景,软件包部署会将构建出来的软件包上传到OBS桶中并升级CAE组件。
- 场景二:使用Jenkins构建生成的是镜像包,就使用脚本中的镜像部署场景,镜像 部署会将构建出来的镜像包上传到SWR镜像仓库中并升级CAE组件。

本章节以配置流水线脚本中的实例为Jar包的场景进行说明。

创建 Gitlab 凭证

使用具有Gitlab代码仓库权限的账号密码在Jenkins中创建凭证,用于拉取Gitlab代码。 步骤1 在浏览器地址栏输入http://{安装Jekins的Linux虚拟机IP}:8080,登录Jenkins。

步骤2选择"系统管理 > 系统配置",在配置中选择"Gitlab"。

图 3-18 进入 Gitlab 配置

| 🍓 Jenki | ns | |
|-------------|--|------------|
| | | |
| Dashboard) | 配置 | |
| | Jenkins Location | |
| 쓸 新建任务 | Serve resource files from another domain | Ł ? |
| A 用户列表 | 全局属性 | jenkins |
| | Pipeline Speed/Durability Settings | |
| 🔁 构建历史 | 使用统计 | 追 ? |
| 0 项目关系 | Folder | |
| | Gitlab | |
| 🏭 检查文件指纹 | Timestamper | |
| the second | Fingerprints | |
| 👷 系統管理 | 管理监控配置 | [4] 预览 |
| 鵗 我的视图 | Global Build Discarders | 中国 |
| | GitHub | 1款里 |
| 💼 新建视图 | GitHub API usage | |
| | GitHub Enterprise Servers | |
| 构建队列 | Generic Webhook Trigger | |
| 队列中没有构建任务 | Global Pipeline Libraries | |
| | Git plugin | |
| 构建执行状态 | Shell | 2 |
| | 即件通知 | V |
| 1 空闲 | | 尽可能的使用这个节点 |
| 2 空闲 | | |

步骤3 单击"Credentials"下方的"添加",选择"Jenkins"。

图 3-19 选择凭据提供者

| b connections | |
|--|---------|
| Connection name | |
| gittab | |
| name for the connection | |
| Sitlab host URL | |
| http://100.85.122.207:8090 | |
| he complete URL to the Gitlab server (e.g. http://gitlab.mydomain.com) | |
| redentials | |
| T | |
| API Token for Gitle Gierkins ed | |
| PI Token for accessing Gitmu | |
| Verifiting Separation From | ſ |
| | |
| | Test Co |
| | |
| | |

步骤4 配置Gitlab账号密码,单击"添加",保存配置。

步骤5 选择"系统管理 > Manage Credentials",查看配置的凭据。 唯一标识在配置流水线脚本中会用到。

图 3-20 管理凭据

| 쯜 新建任务 | 管理Jenkins | | | | |
|-------------|---|---|--|--|--|
| 🌯 用户列表 | | | | | |
| 🔁 构建历史 | Building on the built-in node can be a security issue. You should set up dis | istributed builds. See the documentation. | | | |
| 🔍 项目关系 | | | | | |
| 🏥 检查文件指纹 | 您还没有整置院站得求办面防护,这可能会有安全问题。 更多很思,算多考 <u>这个页面。</u> 您可以在安全区域 脸就漂求伪造在护 未终夜当场的配置。 | | | | |
| 🐡 系统管理 | | | | | |
| 🍓 我的視園 | | | | | |
| 🛅 新建视题 | 系统配置 | | | | |
| 构建队列 へ | | | | | |
| 队列中设有构建任务 | Production of the last intervention | | Proved and the second s | | |
| KARAN GARIN | * * | | | | |
| 19942B11002 | XE | | | | |
| 2 208 | | Manage Credentials Configure credentials | 关援配置 配置凭据的提供者和类型 | | |
| | | | | | |

图 3-21 查看凭据

| 条凭据 | | | |
|--------------------------------------|-------------|-----------|-------------------|
| 类型 提供者 存储 1 | ы | 唯一标识 | 名称 |
| 💽 🕋 Jenkins | 全局 | b12 553 | Git ten |
| 🚨 🅋 Jenkins | 全局 | 133b 3df8 | 2000000614/****** |
| 醫际 小 中 大 Stores scoped to lenkins | | | |
| 提供者 存储 : | Ŀ | | |
| Jenkins | <u>é</u> 全局 | | |
| | | | |

----结束

创建流水线任务

- 步骤1 在浏览器地址栏输入http://{安装Jekins的Linux虚拟机IP}:8080,登录Jenkins。
- 步骤2 单击"新建任务"。

| | 图 3-22 新建任务 | | | | | | |
|-----|--------------------|-------|-----------|-------|------|------|------|
| | 🏘 Jenkin | IS | | | | | |
| | Dashboard > | | | | | | |
| | 쯜 新建任务 | | | | | | |
| | 鶡 用户列表 | | | 所有 | + | | |
| | 🔁 构建历史 | | | | | | |
| 步骤3 | 输入任务名称,本 | 示例使用: | test-demo | ,选择"流 | 水线", | 单击"矸 | 角定"。 |

----结束

配置构建触发器

这里介绍两种构建方式:

1. 在Jenkins中手动触发构建,手动单击任务右边的"立即构建"从而触发流水线任务。

2. 通过Gitlab提交代码后自动触发Jenkins构建,这种方式同时支持手动触发构建。

此处以第二种方式为例。

步骤1 配置Jenkins构建触发器。

1. 勾选"Build when a change is pushed to GitLab",保存图示中的GitLab webhook URL(配置Gitlab webhook时需使用),然后单击右下角"高级"。

图 3-23 获取 GitLab webhook URL

| 构建触发器 | |
|--|--------------|
| □ 貸前工程均建后被发 ? □ 意时接续 ? ■ Bulld when a change is pushed to Gittab Gittab webhook URL: http://********************************** | |
| Push Events in case of branch delete | |
| Opened Merge Request Events | |
| Build only if new commits were pushed to Merge Request (?) | |
| Accepted Merge Request Events | |
| Closed Merge Request Events | |
| Rebuild open Merge Requests | |
| Never | ~ |
| Approved Merge Requests (EE-only) | |
| Comments | |
| Comment (regex) for triggering a build 7 | |
| Jenkins please retry a build | |
| | 2 a g |

2. 选择"Filter branches by regex",配置指定分支变更后触发构建任务,示例中的分支名称为main,单击右下角"Generate"生成Secret token并保存,在配置Gitlab webhook时需使用。

图 3-24 获取 Secret token

| Comment (regex) for triggering a build | | |
|---|-----------------|-----------------------------|
| Jenkins please retry a build | | |
| Enable [ci-skip] | | |
| Ignore WIP Merge Requests | | |
| l skale ábat áranna kullda lá ában ann addad (annuna annantad) | | |
| cabes that forces builds if they are abled (comma-separated) | | |
| | | |
| Set build description to build cause (eg. Merge request or Git) | Push) | |
| Build on successful pipeline events | | |
| Pending build name for pipeline (?) | | |
| | | |
| Cancel pending merge request builds on update | | |
| Allowed branches | | |
| O Allow all branches to trigger this job | 化宁林宁丛士亦再已轴尖构建几夕 | |
| Filter branches by name P Filter branches by regex P | 相足付足力又又更肩胝及构建任务 | |
| Source Branch Regex | | |
| main | | |
| Target Branch Regex | | |
| | | |
| Filter merge request by label | | |
| Secret token ? | | 生成token,配置gitlab webhook 需要 |
| 12fe6a647812ec85bc6d23e81bf9e75e | | |
| | | Generate |
| | | Qear |
| | | |

步骤2 配置Gitlab webhook

- 1. 登录Gitlab,进入代码仓库,示例中的仓库名称是"test-demo"。
- 2. 选择settings中的"Webhooks",URL和Secret token填写步骤1获取到的GitLab webhook URL和Secret token。

图 3-25 webhooks 配置

| 🤟 GitLab 😑 Menu | | | ۰ |
|---|---|--|---|
| T test-demo T Project information Repository | Open registration is enabled on your Learn more about how you can custor View atting | Instance. X | |
| UP Issues 0 11 Merge requests 0 CI/CD | hap + tast-dens - Y Weldwold Settings | | |
| Decurity & Compliance Deployments Monitor Intractructure Packages & Registries Analytics Whit Snippets Sestings | Webhooks Waterooks and a post to seen notification we be application in the second and a part group or project. We accommend using an engingetion in profession to a webhook. | UR: | |
| General Integrations Webhooks Access Tokens Repository CL/CD Monitor | | Branch name or velocited gradients to togger on flavors blank for vit) URL is triggered by a public to the reportiony Care and the second se | |

3. 取消勾选SSL verification的"Enable SSL verification",单击"Add webhook"。

图 3-26 完成 webhook 配置

| SSL verification | |
|-------------------------|--|
| Enable SSL verification | |
| Add webhook | |

----结束

配置流水线脚本

流水线脚本是构建时运行的构建命令,脚本参数说明见表3-3。

表 3-3 流水线脚本参数说明

| 参数 | 是否必 须 | 参数类型 | 描述 |
|--------------------|----------|--------|---|
| git_url | 是 | String | Gitlab代码仓库地址。 |
| credentials _id | 是 | String | 使用账号密码配置的Gitlab凭据id,参考 <mark>创建</mark> <mark>Gitlab凭证</mark> 。 |
| branch_na me | 是 | String | Gitlab代码仓库分支名称。 |
| maven | 是 | String | maven安装的可执行文件路径,示例:/ root/app/maven/apache-maven-3.8.6/bin/ mvn。 |
| deploy_she ll | 是 | String | deploy.sh脚本在Jenkins所在虚拟机上存放的路 径,示例:/root/jar/deploy.sh,内容请参见 deploy.sh脚本说明 。 |
| 参数 | 是否必 须 | 参数类型 | 描述 |
|-----------------------|----------|--------|--|
| build_targe t_name | 是 | String | 构建产物名称:软件包名称或镜像名称:版本 号,通过执行脚本时传入该参数,软件包部署 场景为软件包名称,镜像部署场景为构建出来 的镜像名称:版本号。 |

步骤1 完成"构建触发器"配置之后,在"流水线"页签,在下拉框选择"Pipeline script"。

图 3-27 流水线配置

| <pre>Product void void void void void void void void</pre> | Pipeline | |
|--|---|--|
| <pre>bit ?</pre> | - ipenne s | ript |
| <pre>wt **</pre> | 80.00 | |
| <pre>//***********************************</pre> | 1 - | node (|
| <pre>// def does y de former in and in the second s</pre> | 2 3 4 | // 定义代码包括如 def git unl = "http://100. 9090/hzp/cae-backend.git" // Gitlab提罪d |
| <pre>visition of the second se</pre> | 5 | def credentials_id = '113b7c9a-eb6a-4484-84b3-c3589ed63df8'' //git代符合展分至26冷 |
| <pre>// fight and the set of the</pre> | 8 | Ger bindlef "male" // maveng要的可执行文件最经 def maven = '/root/app/maven/apache-maven-3.8.6/bin/mvn' |
| <pre>// Page Page - State State Page - State - St</pre> | 10 11 12 | // deploy.sh 即本存款局份,需要进版可执行发现 def deploy_shell = '/root/project/cae-backend/deploy.sh' |
| <pre>###('Come server') { formative: // ###</pre> | 13 14 | // 构遗产物名称: 软件包名称成摄像名称,非必须参数,可以通过执行脚本时待入该参数 def build_target_name = "cae-demo-1.0-SMAPSHOT.jan" |
| <pre>if if i</pre> | 16 * 17 | <pre>stage('Clone sources') { git branch: branch_name, credentials_id, un1: git_un1</pre> |
| <pre>if "Berer 'law is the package -beam left.filles.gaver the 'beam left.filles.gaver the 'beam left.' if 'left.'' if 'left.</pre> | 18 19 * 20 |) stage('Bulla') (// 付達:jar包 |
| <pre>if if i</pre> | 21 22 23 - | <pre>sh "'Smaven' clean package -Dmaven.test.fallure.ignore=true -Dmaven.wagon.http.ssl.insecure=true" } ctrum('idmlou') /</pre> |
| <pre>// # The mater with the property and the material of the</pre> | 24 25 | A Gate (Mode) / (人行即本,使用物連上体列Adoe的jar包升板(Adoe目年,超时封同5分時 sh "timeout 300s; Sdoploy shall:" |
| <pre>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>></pre> | 26 27 28 | // F2/代理/ F2/存在存在: #21年日在存在10月中可 //sh "timeout 300s '\$deploy_shell' '\$build_target_name'* } |
| ■ #################################### | 29 |) |
| <pre> @ ##Goovy 201 ? previous 201</pre> | | |
| ##### ## ## ## ## ## ## ## ## | 1963 | Groovy DPA C |
| <pre>wmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm</pre> | 流水线 | ik de la constante de la const |
| <pre>wm wm w</pre> | | |
| 配置流水线脚本,示例中使用的是构建jar包场景,脚本如下: node { // 定义代码仓地址 def git_url = 'http://100.**.**.207:8090/test/test-demo.git' // Gitlab凭据id def credentials_id = '133b7c9a-eb6a-4484-84b3-c3509ed63df8' // git代码仓库分支名称 def branch_name = 'main' // maven安装的可执行文件路径 def maven = '/root/app/maven/apache-maven-3.8.6/bin/mvn' // deploy.sh 脚本存放路径,需要设置可执行权限 def deploy_shell = '/root/jar/deploy.sh' // 构建产物名称: 软件包名称或镜像名称,必须参数,通过执行脚本时传入该参数 def build_target_name = "cae-demo-1.0-SNAPSHOT.jar" stage('Clone sources') { git branch: branch_name, credentialsId: credentials_id, url: git_url } stage('Build') { // 构建jar包 sh ""\$maven' clean package -Dmaven.test.failure.ignore=true" | 保ィ | 应用 |
| 配置流水线脚本,示例中使用的是构建jar包场景,脚本如下: mode { // 定义代码仓地址 def git_url = 'http://100.**.**.207:8090/test/test-demo.git' // Gitlab凭据id def credentials_id = '133b7c9a-eb6a-4484-84b3-c3509ed63df8' // git代码仓库分支名称 def branch_name = 'main' // maven安装的可执行文件路径 def maven = '/root/app/maven/apache-maven-3.8.6/bin/mvn' // deploy.sh 脚本存放路径,需要设置可执行权限 def deploy_shell = '/root/jar/deploy.sh' // 构建产物名称: 软件包名称或镜像名称,必须参数,通过执行脚本时传入该参数 def build_target_name = "cae-demo-1.0-SNAPSHOT.jar" stage('Clone sources') { git branch: branch_name, credentialsId: credentials_id, url: git_url } stage('Build') { // 构建jar包 sh "'\$maven' clean package -Dmaven.test.failure.ignore=true" | | |
| dd 量 /// 分生2 // 动体 / , 小 / 列 / 一 使 / 田 J / 全 / 马 / 全 / 小 本 / 本 / 本 / 本 / 本 / 本 / 本 / 本 / 本 / | 印罢 | 去水线脚木, 于例由估田的旦构建iar包括星, 啣木加下· |
| lode { // 定义代码仓地址 def git_url = 'http://100.**.**.207:8090/test/test-demo.git' // Gitlab凭据id def credentials_id = '133b7c9a-eb6a-4484-84b3-c3509ed63df8' // git代码仓库分支名称 def branch_name = 'main' // maven安装的可执行文件路径 def maven = '/root/app/maven/apache-maven-3.8.6/bin/mvn' // deploy.sh 脚本存放路径,需要设置可执行权限 def deploy_shell = '/root/jar/deploy.sh' // 构建产物名称: 软件包名称或镜像名称,必须参数,通过执行脚本时传入该参数 def build_target_name = "cae-demo-1.0-SNAPSHOT.jar" stage('Clone sources') { git branch: branch_name, credentialsId: credentials_id, url: git_url } stage('Build') { // 构建jar包 sh ""\$maven' clean package -Dmaven.test.failure.ignore=true" | 10月 | |
| <pre>// Lex Total Statustion def git_url = 'http://100.**.**.207:8090/test/test-demo.git' // Gitlab凭据id def credentials_id = '133b7c9a-eb6a-4484-84b3-c3509ed63df8' // git代码仓库分支名称 def branch_name = 'main' // maven安装的可执行文件路径 def maven = '/root/app/maven/apache-maven-3.8.6/bin/mvn' // deploy.sh 脚本存放路径,需要设置可执行权限 def deploy_shell = '/root/jar/deploy.sh' // 构建产物名称: 软件包名称或镜像名称,必须参数,通过执行脚本时传入该参数 def build_target_name = "cae-demo-1.0-SNAPSHOT.jar" stage('Clone sources') { git branch: branch_name, credentialsId: credentials_id, url: git_url } stage('Build') { // 构建jar包 sh ""\$maven' clean package -Dmaven.test.failure.ignore=true"</pre> | a da | ル小地科学,小別中使用型定性度lal 巴切京,脚子知下· |
| // Gitlab凭据id // Gitlab凭据id def credentials_id = '133b7c9a-eb6a-4484-84b3-c3509ed63df8' // git代码仓库分支名称 def branch_name = 'main' // maven安装的可执行文件路径 def maven = '/root/app/maven/apache-maven-3.8.6/bin/mvn' // deploy.sh 脚本存放路径,需要设置可执行权限 def deploy_shell = '/root/jar/deploy.sh' // 构建产物名称: 软件包名称或镜像名称,必须参数,通过执行脚本时传入该参数 def build_target_name = "cae-demo-1.0-SNAPSHOT.jar" stage('Clone sources') { git branch: branch_name, credentialsId: credentials_id, url: git_url } stage('Build') { // 构建jar包 sh ""\$maven' clean package -Dmaven.test.failure.ignore=true" | ode · | ル小ジ科学,小別学社大力型定型建Jal 巴切京,脚子知下。 |
| def credentials_id = '133b7c9a-eb6a-4484-84b3-c3509ed63df8' // git代码仓库分支名称 def branch_name = 'main' // maven安装的可执行文件路径 def maven = '/root/app/maven/apache-maven-3.8.6/bin/mvn' // deploy.sh 脚本存放路径,需要设置可执行权限 def deploy_shell = '/root/jar/deploy.sh' // 构建产物名称: 软件包名称或镜像名称,必须参数,通过执行脚本时传入该参数 def build_target_name = "cae-demo-1.0-SNAPSHOT.jar" stage('Clone sources') { git branch: branch_name, credentialsId: credentials_id, url: git_url } stage('Build') { // 构建jar包 sh ""\$maven' clean package -Dmaven.test.failure.ignore=true" | ode // 5 | ルルンジョッキ、ハロッキロオロンをでみ生jal 巴切京、阿平知I ト・ ミ ミン代码仓地址 ait ud = 'http://100 ** ** 207:8090/test/test-demo.ait' |
| <pre>// git代码仓库分支名称 def branch_name = 'main' // maven安装的可执行文件路径 def maven = '/root/app/maven/apache-maven-3.8.6/bin/mvn' // deploy.sh 脚本存放路径,需要设置可执行权限 def deploy_shell = '/root/jar/deploy.sh' // 构建产物名称: 软件包名称或镜像名称,必须参数,通过执行脚本时传入该参数 def build_target_name = "cae-demo-1.0-SNAPSHOT.jar" stage('Clone sources') { git branch: branch_name, credentialsId: credentials_id, url: git_url } stage('Build') { // 构建jar包 sh "'\$maven' clean package -Dmaven.test.failure.ignore=true"</pre> | ode // テ def // (| ルットシューキ、ハロッサロナロン定で到生Jal 巴切京、阿平知IF・ E义代码仓地址 git_url = 'http://100.**.**.207:8090/test/test-demo.git' itlab凭据id |
| def branch_name = 'main' // maven安装的可执行文件路径 def maven = '/root/app/maven/apache-maven-3.8.6/bin/mvn' // deploy.sh 脚本存放路径,需要设置可执行权限 def deploy_shell = '/root/jar/deploy.sh' // 构建产物名称: 软件包名称或镜像名称,必须参数,通过执行脚本时传入该参数 def build_target_name = "cae-demo-1.0-SNAPSHOT.jar" stage('Clone sources') { git branch: branch_name, credentialsId: credentials_id, url: git_url } stage('Build') { // 构建jar包 sh ""\$maven' clean package -Dmaven.test.failure.ignore=true" | ode // テ def // C def | ルットンジャット、ハンジリナービナガロンティンシェ」ロービック京 、 阿本文 XH 「・ ミン代码仓地址 git_url = 'http://100.**.**.207:8090/test/test-demo.git' iitlab凭据id credentials_id = '133b7c9a-eb6a-4484-84b3-c3509ed63df8' |
| <pre>// maven女装担可执行又件路径 def maven = '/root/app/maven/apache-maven-3.8.6/bin/mvn' // deploy.sh 脚本存放路径,需要设置可执行权限 def deploy_shell = '/root/jar/deploy.sh' // 构建产物名称: 软件包名称或镜像名称,必须参数,通过执行脚本时传入该参数 def build_target_name = "cae-demo-1.0-SNAPSHOT.jar" stage('Clone sources') { git branch: branch_name, credentialsId: credentials_id, url: git_url } stage('Build') { // 构建jar包 sh ""\$maven' clean package -Dmaven.test.failure.ignore=true"</pre> | ode // 5 def // 0 def // 0 | ルパンシルイン・パッサービルロン定行到建Jai 巴切京, 阿平XII ト・ ミン代码仓地址 git_url = 'http://100.**.**.207:8090/test/test-demo.git' iitlab凭据id credentials_id = '133b7c9a-eb6a-4484-84b3-c3509ed63df8' it代码仓库分支名称 |
| <pre>// deploy.sh 脚本存放路径,需要设置可执行权限 def deploy_shell = '/root/jar/deploy.sh' // 构建产物名称: 软件包名称或镜像名称,必须参数,通过执行脚本时传入该参数 def build_target_name = "cae-demo-1.0-SNAPSHOT.jar" stage('Clone sources') { git branch: branch_name, credentialsId: credentials_id, url: git_url } stage('Build') { // 构建jar包 sh "'\$maven' clean package -Dmaven.test.failure.ignore=true"</pre> | ode // 5 def // 0 def // 0 def | ルバンシスローク、パワリーロンモロンモロシスETタンEJal 巴切京、阿平文知下・ ミン代码仓地址 git_url = 'http://100.**.**.207:8090/test/test-demo.git' iitlab凭据id credentials_id = '133b7c9a-eb6a-4484-84b3-c3509ed63df8' iit代码仓库分支名称 branch_name = 'main' |
| <pre>// 站中达Jian Wether Backaller, mgg ge Backaller, ba</pre> | ode // 5 def // 0 def // 0 def // r | ルバンシスローチ、バンドッサービナガロンディタメ生Jal ビ切京、 阿子XIト・ = ミン代码仓地址 git_url = 'http://100.**.**.207:8090/test/test-demo.git' itlab凭据id credentials_id = '133b7c9a-eb6a-4484-84b3-c3509ed63df8' it代码仓库分支名称 branch_name = 'main' haven安装的可执行文件路径 mayen_= '/root/ann/mayen/apache_mayen_3.8.6/bin/myn' |
| <pre>// 构建产物名称:软件包名称或镜像名称,必须参数,通过执行脚本时传入该参数 def build_target_name = "cae-demo-1.0-SNAPSHOT.jar" stage('Clone sources') { git branch: branch_name, credentialsId: credentials_id, url: git_url } stage('Build') { // 构建jar包 sh "'\$maven' clean package -Dmaven.test.failure.ignore=true"</pre> | ode // 5 def // 0 def // 0 def // r def // r | ルリンニンBAY 44、ハバアリヤ 1 史 H ロン 天 T タメモ J ロ ビ 切 京 , |
| def build_target_name = "cae-demo-1.0-SNAPSHOT.jar" stage('Clone sources') { git branch: branch_name, credentialsId: credentials_id, url: git_url } stage('Build') { // 构建jar包 sh ""\$maven' clean package -Dmaven.test.failure.ignore=true" | iode // 5 def // 0 def // 0 def // r def // c def | ルバンシルロイン・バアリー・ビアドロン定て少生Jai ビの京 , 麻(小)、ビアド ジ代码合地址 git_url = 'http://100.**.**.207:8090/test/test-demo.git' itlab凭据id credentials_id = '133b7c9a-eb6a-4484-84b3-c3509ed63df8' it代码仓库分支名称 branch_name = 'main' haven安装的可执行文件路径 maven = '/root/app/maven/apache-maven-3.8.6/bin/mvn' eploy.sh 脚本存放路径,需要设置可执行权限 deploy_shell = '/root/jar/deploy.sh' |
| stage('Clone sources') { git branch: branch_name, credentialsId: credentials_id, url: git_url } stage('Build') { // 构建jar包 sh ""\$maven' clean package -Dmaven.test.failure.ignore=true" | ode // 5 def // 0 def // r def // r def // c def // c | ルレス・シルムイキ、 ハバッサービナカロン 定て少ま」ロ ビ 切 京 , |
| stage('Clone sources') { git branch: branch_name, credentialsId: credentials_id, url: git_url } stage('Build') { // 构建jar包 sh ""\$maven' clean package -Dmaven.test.failure.ignore=true" } | def // 5 def // 0 def // c def // r def // c def // t def | ルレス・シルムイキ、 ハバウリヤ ビア オロン 定 イタメ主 J ロ ビ 切 京 、 麻レイト メロ ト・ ご ご ご ご ご ご ご し い で た て r の に し に い こ 3 5 7 5 3 5 5 5 5 5 5 5 5 5 5 5 5 5 |
| git branch_branch_hame, credentialsid: credentials_id, urt: git_urt } stage('Build') { // 构建jar包 sh ""\$maven' clean package -Dmaven.test.failure.ignore=true" | ode // 5 def // 0 def // r def // r def // r def // r | ルレス・シルム・キ、ハバウリヤリナハロンズモイタメ主人は「ビンジステ、「麻ノキンメロト・ ごと代码仓地址 git_url = 'http://100.**.**.207:8090/test/test-demo.git' iitlab凭据id credentials_id = '133b7c9a-eb6a-4484-84b3-c3509ed63df8' iit代码仓库分支名称 branch_name = 'main' haven安装的可执行文件路径 maven = '/root/app/maven/apache-maven-3.8.6/bin/mvn' eploy.sh 脚本存放路径,需要设置可执行权限 deploy_shell = '/root/jar/deploy.sh' D建产物名称: 软件包名称或镜像名称,必须参数,通过执行脚本时传入该参数 build_target_name = "cae-demo-1.0-SNAPSHOT.jar" |
| stage('Build') { // 构建jar包 sh "'\$maven' clean package -Dmaven.test.failure.ignore=true" | ode // 5 def // 0 def // r def // r def // r def // k | ルレスシニルロ44、ハバワリヤロンモロ9年J&I 色切京, 麻44×41 ト・ E义代码仓地址 git_url = 'http://100.**.**.207:8090/test/test-demo.git' iitlab凭据id credentials_id = '133b7c9a-eb6a-4484-84b3-c3509ed63df8' iit代码仓库分支名称 branch_name = 'main' haven安装的可执行文件路径 maven = '/root/app/maven/apache-maven-3.8.6/bin/mvn' eploy.sh 脚本存放路径,需要设置可执行权限 deploy_shell = '/root/jar/deploy.sh' 9建产物名称: 软件包名称或镜像名称,必须参数,通过执行脚本时传入该参数 build_target_name = "cae-demo-1.0-SNAPSHOT.jar" pe('Clone sources') { |
| // 构建jar包 sh "'\$maven' clean package -Dmaven.test.failure.ignore=true" | ode - // 5 def // 0 def // r def // r def // r def // t def | ルレスシェルロイキ、「ハドリーヤービナトロン定て少生」は「ビック京 、「レイキタロト・ E义代码仓地址 git_url = 'http://100.**.**.207:8090/test/test-demo.git' iitlab凭据id credentials_id = '133b7c9a-eb6a-4484-84b3-c3509ed63df8' iit代码仓库分支名称 branch_name = 'main' haven安装的可执行文件路径 maven = '/root/app/maven/apache-maven-3.8.6/bin/mvn' eploy.sh 脚本存放路径,需要设置可执行权限 deploy_shell = '/root/jar/deploy.sh' g建产物名称: 软件包名称或镜像名称,必须参数,通过执行脚本时传入该参数 build_target_name = "cae-demo-1.0-SNAPSHOT.jar" ge('Clone sources') { jit branch: branch_name, credentialsId: credentials_id, url: git_url |
| sh "'\$maven' clean package -Dmaven.test.failure.ignore=true" | iode - // 5 def // C def // c def // r def // r def // r def // c def // c def // c state } state } state } | ルレスシミルロイキ、「ハドリーヤ」ビナガロン定て少生」は「ビック京 、「レイキメロト・ ジノ代码仓地址 git_url = 'http://100.**.**.207:8090/test/test-demo.git' iitlab凭据id credentials_id = '133b7c9a-eb6a-4484-84b3-c3509ed63df8' it代码仓库分支名称 branch_name = 'main' haven安装的可执行文件路径 maven = '/root/app/maven/apache-maven-3.8.6/bin/mvn' eploy.sh 脚本存放路径,需要设置可执行权限 deploy_shell = '/root/jar/deploy.sh' g建产物名称: 软件包名称或镜像名称,必须参数,通过执行脚本时传入该参数 build_target_name = "cae-demo-1.0-SNAPSHOT.jar" ge('Clone sources') { jit branch: branch_name, credentialsId: credentials_id, url: git_url me('Build') { |
| | ode - // 5 def // 0 def // r def // r def // r def // r def // sta | ルレス・シルムイキ、ハ、ドッサービナガロンズモイタメ生Jロビンの京 、 麻レイキメロ ト・ ジノ代码仓地址 git_url = 'http://100.**.**.207:8090/test/test-demo.git' iitlab凭据id credentials_id = '133b7c9a-eb6a-4484-84b3-c3509ed63df8' it代码仓库分支名称 branch_name = 'main' haven安装的可执行文件路径 maven = '/root/app/maven/apache-maven-3.8.6/bin/mvn' eploy.sh 脚本存放路径,需要设置可执行权限 deploy_shell = '/root/jar/deploy.sh' p建产物名称: 软件包名称或镜像名称,必须参数,通过执行脚本时传入该参数 build_target_name = "cae-demo-1.0-SNAPSHOT.jar" ge('Clone sources') { jit branch: branch_name, credentialsId: credentials_id, url: git_url pe('Build') { / 构建jar包 |
| } | ode // ½ def // c def // r def // r def // r def // c def // z sta | nL/J、キ&BAV44、J、バアリサイビナガロン定て少生Jのビリク京 , MV44XU ト・ ジノ代码仓地址 git_url = 'http://100.**.**.207:8090/test/test-demo.git' iitlab凭据id credentials_id = '133b7c9a-eb6a-4484-84b3-c3509ed63df8' it代码仓库分支名称 branch_name = 'main' haven安装的可执行文件路径 maven = '/root/app/maven/apache-maven-3.8.6/bin/mvn' eploy.sh 脚本存放路径,需要设置可执行权限 deploy_shell = '/root/jar/deploy.sh' p建产物名称: 软件包名称或镜像名称,必须参数,通过执行脚本时传入该参数 build_target_name = "cae-demo-1.0-SNAPSHOT.jar" ge('Clone sources') { jit branch: branch_name, credentialsId: credentials_id, url: git_url pe('Build') { / 构建jar包 h "'\$maven' clean package -Dmaven.test.failure.ignore=true" |
| | ode // ½ def // c def // c def // r def // r def // r def // sta } sta | NL/J、2×8444、、J、I/Jリーロンモロシモロシテロシテ、MJ-4×94下・ 22代码仓地址 git_url = 'http://100.**.**.207:8090/test/test-demo.git' iitlab凭据id credentials_id = '133b7c9a-eb6a-4484-84b3-c3509ed63df8' iit代码仓库分支名称 branch_name = 'main' haven安装的可执行文件路径 maven = '/root/jap/maven/apache-maven-3.8.6/bin/mvn' eploy.sh 脚本存放路径,需要设置可执行权限 deploy_shell = '/root/jar/deploy.sh' D建产物名称: 软件包名称或镜像名称,必须参数,通过执行脚本时传入该参数 build_target_name = "cae-demo-1.0-SNAPSHOT.jar" pe('Clone sources') { jit branch: branch_name, credentialsId: credentials_id, url: git_url pe('Build') { / 构建jar包 h "'\$maven' clean package -Dmaven.test.failure.ignore=true" pe('deploy') { |

```
// 执行脚本,使用构建产物升级CAE组件,超时时间5分钟
sh "timeout 300s '$deploy_shell' '$build_target_name'"
}
}
```

🛄 说明

- 流水线脚本运行时会调用deploy.sh,该脚本详细说明参见deploy.sh脚本说明。
- 设置脚本文件deploy.sh为可执行文件。

----结束

#!/bin/bash

3.3.3 deploy.sh 脚本说明

使用场景说明

- 场景一:使用Jenkins构建生成的是软件包,如jar包,就使用脚本中的软件包部署场景,软件包部署会将构建出来的软件包上传到obs桶中,再使用新的jar包去升级CAE组件。
- 场景二:使用Jenkins构建生成的是镜像,就使用脚本中的镜像部署场景,镜像部 署会将构建出来的镜像上传到swr镜像仓库中,再使用新的镜像去升级CAE组件。

脚本内容

| # |
|---|
| # region名称 |
| region=" |
| # 坝目id |
| project_id=" |
| # 环境名称 |
| environment_name=" |
| |
| application_name=" |
| |
| component_name="""" + 如果类型。1.coftures2.impege_coftures主一物供有效果impege主一倍值效果 |
| # Ph者关空. 1. Software 2. Image Software农小软件也하者, Image农小说像하者 |
| deptoy_type=" |
| # 如此有刘玺招星心情会粉 |
| #~ |
| |
| # 左対軟件句的obs桶名和 |
| " TTAXATTESUSUSTATION |
| |
| |
| Live dir='/' |
| |
| # |
| # swrf组织名称 |
| swr organization=" |
| # AK 用于登录swr镜像仓库 |
| AK=" |
| # swr登录密钥 用于登录swr镜像仓库 |
| login secret=" |
| |
| # |
| |
| # 软件包部署场景:构建出来的软件包名称,由外部build_target_name参数传入 |
| software_package=" |
| # 镜像部署场景:构建后生成的镜像:镜像名称:版本名称,由外部build_target_name参数传入 |
| machine_image_name=" |
| |

```
#必须传入一个参数,是软件包名称或镜像名称
#参数数量为1时,设置软件包名称或镜像名称
if [ $# == 1 ]; then
  if [[ "$deploy_type" == "software" ]]; then
    software_package=$1
  elif [[ "$deploy_type" == "image" ]]; then
    machine_image_name=$1
  fi
else
  echo "The number of external input parameters is not 1. Configure the pipeline script parameter
build_target_name in the pipeline build task."
fi
enterprise_project_id='0'
environment id='
application_id="
component_id="
function splitVersion() {
  str=$1
  right=${str##*.}
  left=${str%.*}
  if [ ${#str} -eq ${#left} ]; then
    left=""
  fi
}
function addVersion() {
  v1=$1
  res=`echo $v1|awk '{print int($0)}'`
  if [ $res -gt 9998 ]; then
    res=0
  else
    res=$(($res+1))
  fi
}
# 升级版本号
#版本号格式为:A.B.C或A.B.C.DA、、B、C、D均为整数,每个整数不大于9999
# 示例: 1.0.0 -> 1.0.1; 1.0.9999 -> 1.1.0
function upgradeVersion() {
  left=$1
  start=0
  end="
  res=0
  while ( [[ $res = 0 ]] && [ $start -lt 4 ] )
  do
    start=$(($start+1))
    splitVersion $left
    addVersion $right
    if [ $start -gt 1 ]; then
       end=`echo "$res.$end"`
    else
       end=`echo "$res$end"`
    fi
  done
  if [ ${#left} -ne 0 ]; then
    version="$left.$end"
  else
    version="$end"
  fi
}
#获取组件升级的版本号
function upgradeComponentVersion() {
  # 查询组件信息
  component detials=`/usr/local/bin/hcloud CAE ShowComponent --cli-region="$region" --
project_id="$project_id" --X-Enterprise-Project-ID="$enterprise_project_id" --X-Environment-
```

```
ID="$environment_id" --application_id="$application_id" --component_id="$component_id"`
  # 获取组件当前的版本号
  version=`getJsonValuesByAwk "$component_detials" "version" "defaultValue"`
  version=`echo "$version" | cut -d "" -f 2`
  echo "current version: $version"
  if [[ "$version" == "defaultValue" ]]; then
    echo "get component detials and get component version error"
    echo "$component_detials"
    exit 126
  fi
  upgradeVersion $version
  echo "upgrade version: $version"
# 软件包部署使用场景
function obs_software_upgrade() {
  echo "upload jar to obs'
  # 软件包上传到obs的地址,格式: obs://{桶名}{jar包在obs桶中的存放目录}
  bucket_address="obs://$bucket_name$bucket_dir"
  #上传到obs的jar包链接,格式:https://{桶名}.obs.{region}.myhuaweicloud.com/{jar包在obs桶中的存放目
录}/
  obs_jar_url="https://$bucket_name.obs.$region.myhuaweicloud.com$bucket_dir$software_package"
  echo "software_package url: $obs_jar_url"
  # 将项目下构建生成的jar包上传到obs
  obs_result=`"$obsutil" cp ./target/*.jar "$bucket_address"`
  if [ $? -ne 0 ]; then
    echo "obsutil upload software package error"
    echo "$obs_result"
    exit 126
  fi
  # 打印上传结果
  echo "$obs_result"
  # 升级组件
  echo "upgrade component"
  action result=`/usr/local/bin/hcloud CAE ExecuteAction --cli-region="$region" --project id="$project id" --
X-Enterprise-Project-ID="$enterprise_project_id" --X-Environment-ID="$environment_id" -
application_id="$application_id" --component_id="$component_id" --api_version='v1' --kind='Action' --
metadata.name='upgrade' --metadata.annotations.version="$version" --spec.source.type='softwarePackage'
--spec.source.sub_type='BinObs' --spec.source.url="$obs_jar_url"`
}
# 生成秒级时间格式字符串,用于上传SWR镜像时,在版本后添加时间格式后缀
function generateTimeFormatString() {
  ms=$(date +%s)
  time=$(date -d @$ms "+%Y%m%d%H%M%S")
  echo $time
}
# 镜像部署使用场景
function swr_image_upgrade() {
  # 上传到swr的镜像仓库路径,格式为: [镜像仓库地址]/[组织名称]/[镜像名称:版本名称]
  swr_image_url="swr.$region.myhuaweicloud.com/$swr_organization/$machine_image_name"
  # swr 镜像仓库地址
  swr_url="swr.$region.myhuaweicloud.com"
  # swr镜像版本拼接时间格式字符串
  time_format_string=`generateTimeFormatString`
  swr_image_url="$swr_image_url.$time_format_string"
  echo "upload image to swr"
  docker tag "$machine_image_name" "$swr_image_url"
  login_result=`docker login -u "$region"@"$AK" -p "$login_secret" "$swr_url"`
  #打印登录swr镜像仓库结果
  #echo "$login_result"
  push_result=`docker push "$swr_image_url"`
  if [ $? -ne 0 ]; then
```

```
echo "docker push error"
    echo "$push_result"
     exit 126
  fi
  # 打印推送镜像结果
  #echo "$push_result"
  logout_result=`docker logout "$swr_url"`
  # 打印退出登录swr镜像仓库的结果
  #echo "$logout_result"
  # 清除所有的历史记录,历史记录中可能会存在swr登录密钥信息,可以使用该命令清理所有的历史记录
  #history -c
  echo "upgrade component"
  #升级 image镜像组件
  action_result=`/usr/local/bin/hcloud CAE ExecuteAction --cli-region="$region" --project_id="$project_id" --
X-Enterprise-Project-ID="$enterprise_project_id" --X-Environment-ID="$environment_id" -
application_id="$application_id" --component_id="$component_id" --api_version='v1' --kind='Action' --
metadata.name='upgrade' --metadata.annotations.version="$version" --spec.source.type='image' --
spec.source.url="$swr_image_url"
### 方法简要说明:
### 1. 是查找一个字符串:带双引号的key。如果没找到,则直接返回defaultValue。
### 2. 查找最近的冒号,找到后认为值的部分开始了。
### 3. 如果有多个同名key,只打印第一个value
### 4. params: json, key, defaultValue
function getJsonValuesByAwk() {
  awk -v json="$1" -v key="$2" -v defaultValue="$3" 'BEGIN{
    foundKeyCount = 0
     pos = match(json, "\""key"\"[ \\t]*?:[ \\t]*");
     if (pos == 0) {if (foundKeyCount == 0) {print defaultValue;} exit 0;}
     ++foundKeyCount;
     start = 0; stop = 0; layer = 0;
    for (i = pos + length(key) + 1; i <= length(json); ++i) {</pre>
       lastChar = substr(json, i - 1, 1)
       currChar = substr(json, i, 1)
       if (start <= 0) {
         if (lastChar == ":") {
            start = currChar == " " ? i + 1: i;
            if (currChar == "{" || currChar == "[") {
               layer = 1;
            }
         }
       } else {
         if (currChar == "{" || currChar == "[") {
            ++layer;
         if (currChar == "}" || currChar == "]") {
            --layer;
         if ((currChar == "," || currChar == "}" || currChar == "]") && layer <= 0) {
            stop = currChar == "," ? i : i + 1 + layer;
            break;
         }
       }
     if (start <= 0 || stop <= 0 || start > length(json) || stop > length(json) || start >= stop) {
       if (foundKeyCount == 0) {print defaultValue;} exit 0;
    } else {
       print substr(json, start, stop-start);
    }
  }'
}
### 方法简要说明:
###
     从list返回的结果中,找到key=value时的tagKey值
###
      查找json字符串"'key":"value"'',找到后,以这个位置为结尾
###
      查找{最后一次出现的位置lastIndex,再以lastIndex为起始点,查找tagKey,获取tagKey的值
###
### params: json, key, value, tagKey, defaultValue
```

```
function getJsonValuesByAwkWithConditions() {
  awk -v json="$1" -v key="$2" -v value="$3" -v tagKey="$4" -v defaultValue="$5" 'BEGIN{
     foundKeyCount = 0
     pos = match(json, "\""key"\"[ \\t]*?:[ \\t]*\""value"\"");
     if (pos == 0) {print defaultValue; exit 0;}
     str1 = substr(json, 0, pos)
     lastIndex = 0
     while (match(str1, "{")) {
        lastIndex = lastIndex + RSTART
        str1 = substr(str1, RSTART + RLENGTH)
     if (lastIndex == 0) {print defaultValue; exit 0;}
     newStr = substr(json, lastIndex)
     pos1 = match(newStr, "\""tagKey"\"[ \\t]*?:[ \\t]*");
     if (pos1 == 0) {print defaultValue; exit 0;}
     ++foundKeyCount;
     start = 0; stop = 0; layer = 0;
     for (i = pos1 + length(tagKey) + 1; i <= length(newStr); ++i) {
        lastChar = substr(newStr, i - 1, 1)
        currChar = substr(newStr, i, 1)
        if (start <= 0) {
           if (lastChar == ":") {
             start = currChar == " " ? i + 1: i;
              if (currChar == "{" || currChar == "[") {
                layer = 1;
             }
           }
        } else {
           if (currChar == "{" || currChar == "[") {
              ++layer;
           if (currChar == "}" || currChar == "]") {
              --layer;
           if ((currChar == "," || currChar == "}" || currChar == "]") && layer <= 0) {
              stop = currChar == "," ? i : i + 1 + layer;
              break;
          }
        }
     if (start <= 0 || stop <= 0 || start > length(newStr) || stop > length(newStr) || start >= stop) {
        if (foundKeyCount == 0) {print defaultValue;} exit 0;
     } else {
        print substr(newStr, start, stop-start);
     }
  }'
}
# 获取enterprise_project_id environment_id application_id component_id
function initParpare() {
  # 获取environment_id
  listEnvsResult=`/usr/local/bin/hcloud CAE ListEnvironments --cli-region="$region" --
project_id="$project_id"`
  environment_id=`getJsonValuesByAwkWithConditions "$listEnvsResult" "name" "$environment_name"
"id" "defaultValue"
  environment_id=`echo "$environment_id" | cut -d '"' -f 2`
  echo "environment_id: $environment_id"
  if [[ "$environment_id" == "defaultValue" ]]; then
     echo "list environments and get environment_id error"
     echo "$listEnvsResult"
     exit 126
  fi
  # 获取enterprise_project_id
  enterprise_project_id=`getJsonValuesByAwk "$listEnvsResult" "enterprise_project_id" "defaultValue"`
  enterprise_project_id=`echo "$enterprise_project_id" | cut -d '"' -f 2
  echo "enterprise_project_id: $enterprise_project_id"
  if [[ "$enterprise_project_id" == "defaultValue" ]]; then
     echo "get enterprise_project_id error"
```

```
echo "$listEnvsResult"
     exit 126
  fi
  # 获取application_id
  listAppsResult=`/usr/local/bin/hcloud CAE ListApplications --cli-region="$region" --
project_id="$project_id" --X-Environment-ID="$environment_id"
  application_id=`getJsonValuesByAwkWithConditions "$listAppsResult" "name" "$application_name" "id"
'defaultValue"
  application_id=`echo "$application_id" | cut -d "" -f 2`
  echo "application_id: $application_id"
if [[ "$application_id" == "defaultValue" ]]; then
     echo "list applications and get application_id error"
     echo "$listAppsResult"
     exit 126
  fi
  # 获取component_id
  listComponentsResult=`/usr/local/bin/hcloud CAE ListComponents --cli-region="$region" --
project_id="$project_id" --X-Environment-ID="$environment_id" --application_id="$application_id"`
  component_id=`getJsonValuesByAwkWithConditions "$listComponentsResult" "name"
'$component_name" "id" "defaultValue"
  component_id=`echo "$component_id" | cut -d "" -f 2`
  echo "component_id: $component_id"
  if [[ "$component_id" == "defaultValue" ]]; then
     echo "list components and get component_id error"
     echo "$listComponentsResult"
     exit 126
  fi
}
# 每隔15秒查询一次job状态,直到job完成
function waitDeployFinish() {
  sleep 10s
  id="$1"
  leni=${#id}
  id=${id:1:leni-2}
  echo "job_id= $id"
  job_status=""
  while [[ "$job_status" != "success" ]]; do
     job_status_result=`/usr/local/bin/hcloud CAE ShowJob --cli-region="$region" --project_id="$project_id"
--job_id="$id" --X-Environment-ID="$environment_id"
     job_status=`getJsonValuesByAwk "$job_status_result" "status" "defaultValue"`
     if [[ "$job_status" == "defaultValue" ]]; then
        echo "ShowJob failed"
        echo "$job_status_result"
        return
     fi
     job_status=`echo "$job_status" | cut -d "" -f 2`
if [[ "$job_status" != "running" && "$job_status" != "success" ]]; then
        echo 'upgrade failed'
        echo "$job_status_result"
        return
     fi
     sleep 15s
  done
  echo 'upgrade success'
function deploy() {
  if [[ "$deploy_type" == "software" ]]; then
     obs_software_upgrade
  elif [[ "$deploy_type" == "image" ]]; then
     swr_image_upgrade
  else
     return
  fi
  # 打印升级组件的结果
```

```
echo "upgrade result: $action_result"

# 获取结果中的job_id

job_id=`getJsonValuesByAwk "$action_result" "job_id" "defaultValue"`

if [[ "$job_id" == "defaultValue" ]]; then

echo "upgrade failed"

exit 126

fi

# 等待升级完成

waitDeployFinish "$job_id"

}

function main() {

initParpare

upgradeComponentVersion

deploy

}

main
```

脚本参数说明

| 参数 | 是否必须 | 参数 类型 | 描述 |
|----------------------|------|------------|--|
| region | 是 | Stri ng | 项目名称。 |
| project_id | 是 | Stri ng | 项目id。 |
| environme nt_name | 是 | Stri ng | 组件的环境名称。 |
| application _name | 是 | Stri ng | 组件的应用名称。 |
| component _name | 是 | Stri ng | 组件名称。 |
| deploy_typ e | 是 | Stri ng | 部署类型: software或image。 • software表示软件包部署。 • image表示镜像部署。 |
| obsutil | 否 | Stri ng | 当使用软件包部署如jar包部署时为必须参数。上传jar包 到obs的工具安装的绝对路径。 示例:/root/tools/obsutil/obsutil_linux_amd64_5.4.6/ obsutil。 |
| bucket_na me | 否 | Stri ng | 当使用软件包部署时为必须参数。存放软件包的obs桶 名称。 |

表 3-4 脚本参数说明

| 参数 | 是否必须 | 参数 类型 | 描述 |
|----------------------|------|------------|---|
| bucket_dir | 石 | Stri ng | 当使用软件包部署时为必须参数。 软件包在obs桶中的存放目录,默认是根目录,目录需 要以/结尾,如果obs桶中没有这个目录,会自动创建出 该目录。如根目录"/",根目录下面的test目录"/ test/"。 |
| swr_organi zation | 俗 | Stri ng | 当使用镜像部署时为必须参数。上传到swr的组织名 称。 |
| AK | 柘 | Stri ng | 当使用镜像部署时为必须参数。创建永久AK、SK中的 AK参数,用于登录swr镜像仓库。 |
| login_secre t | 否 | Stri ng | 当使用镜像部署时为必须参数。swr的登录密钥,用于 登录swr镜像仓库。通过创建永久AK、SK获取的 AK/SK,执行如下命令,返回的结果就是登录密钥: printf " <i>\$AK</i> " openssl dgst -binary -sha256 -hmac " <i>\$SK</i> " od -An - vtx1 sed 's/[\n]//g' sed 'N;s/\n// |

参数值获取

获取region、project_id值:
 登录CAE控制台,在右上角个人账号上,单击"我的凭证",查看所属区域的项目和项目ID,即为对应的region和project_id值。

图 3-28 我的凭证

| Q | 费用与成本 | 资源 | 企业 | 开发工具 | 备案 | 支持与服务 | 工单 | 简体中文 | | | Þ | ו בּל |
|----------|-------|----|----|------|----|-------|----|----------|-----|-------|---|-------|
| | | | | | | | | 基本信 | 息 | 已实名认证 | | |
| | | | | | | | | 安全设 | 置 | | | × |
| | | | | | | | | 我的凭 | 证 | | | ~ |
| | | | | | | | | 统一身 | 份认证 | | | |
| | | | | | | | | 切换角 | 色 | | | |
| 32d11ffa | 8 🗇 | | | | | | | 标签管 | 理 | | | |
| | | | | | | | | 操作日 | 志 | | | |
| | | | | | | | | | 退 | 出 | - | Q |
| | | | | | | | | | | | | - |

图 3-29 项目列表

| 项目列表 | | |
|--|-------------------|----------------|
| 项目ID 1Ξ | 項目 1三 | 所属区域 1三 |
| | cn-northeast-1 | cn-northeast-1 |
| 4 project_id | cn-north-1 region | 华北-北原— |
| + 7 ********************************** 42 | cn-north-4 | 华北-北京四 |

• 获取组件所在的环境名称environment_name,应用名称application_name和组件名称component_name:

登录CAE控制台,单击"组件列表",找到目标组件,例如image组件,如<mark>图3-30</mark> 所示。

图 3-30 查看组件信息

| 云应用引擎 | 〒 环境: ❶ test ▼ ④ 廠 应用: ap | p • + | |
|----------------------|---|-------------------|------|
| 概览 组件列表 | environment_name 新婚組件 | application_name | |
| 实例列表 | 名称/版本/ID ≑ 代码源 ▽ | 状态 🍞 实例个数 (可用/全部) | 访问地址 |
| 组件配置 组件事件 | test-521 v1.0.0 47df1d0e-d4c9 component name | ∂ 运行中 1/1 | - |
| 组件监控 组件日志 至统设置 | test0410 現象 v1.0.0 現象 8619c992-f788 nginx stable-perl | 未純器 前往配置 0/1 | - |

3.4 构建验证

3.4.1 手动构建验证

- 步骤1 在浏览器地址栏输入http://{安装Jekins的Linux虚拟机IP}:8080,登录Jenkins。
- 步骤2 左侧导航栏单击"我的视图"。
- 步骤3 选择对应的构建任务,单击构建任务名称进入详情界面,示例为"test-demo"。
- **步骤4** 单击右侧 [▷],在左侧构建执行状态中会生成一个构建任务,单击任务编号,选择 "Console Output",如<mark>图3-31</mark>,查看构建输出日志。

图 3-31 手动构建验证

| Dashboard 🔸 admin | 我的视图 月 | 所有 , | | | | | | | |
|-------------------|---|-------------|-----|-----------|-----------|---------|----------------|----------------|-------------------|
| 쯜 新建任务 | | | | | | | | | / 添加说用 |
| 鶡 用户列表 | | 所有 | + | | | | | | |
| ≥ 构建历史 | | | | | | | | | |
| 🔍 项目关系 | | s | w | 名称: | 上次成功 | 上次失败 | 上次持续 | 賣时间 | |
| 检查文件指纹 | | \odot | :ộ: | test-demo | 4分49秒 #74 | 50分 #67 | 1分25 | 秒 | \triangleright |
| 計 新建規图 | | 图标: 小 | , ф | * | | 18104 | 入 Atom feed 全部 | 入 Atom feed 失败 | 入 Atom feed 最新的构成 |
| 构建队列 | ^ | | | | | | | | |
| (列中没有构建任务 | | | | | | | | | |
| 构建执行状态 | ^ | | | | | | | | |
| test-demo | #75 ¥ | | | | | | | | |
| | | | | | | | | | |
| 2 空闲 | ∠ 交更历史 | _ | | | | | | | |
| 2 空闲 | Console Cut | iut | | | | | | | |
| 2 空闲 | ● 安美历史 ■ Console Outp ● 機構物建信意 ◆ Git Build Dat | rut | | | | | | | |

步骤5 查看构建输出日志,构建出来的是jar包,部署成功。

图 3-32 构建输出日志

| "parameters": { "base image": "100.79.1.215:20202/op.svc_cse/open;db@={arch}:1.0.5" |
|--|
| |
| 1, |
| "resource_limit": { |
| "cpu_linit": "500m". |
| "nemory_limit": "IGI" |
| L |
| "access_info": [], |
| "image_url": "swr. cm-morth-7. nyhuaweicloud. com/; I/test-jar:v4", |
| available_replica: 1. |
| job_14 Th7b1196-29r6-41r4-9321-bc2508802968 ⁻ , |
| bulla_la yydagudetipylisylma(chsiutizvead4ewsigvns. |
| status : rumning , |
| purid'rof'rd : |
| |
| J and the second s |
| CHARTER VELOCIES AND A CONTRACT OF |
| unerada vareios: 1.0.3 |
| who ad in to abs |
| [] 100,00% 7/s 31,9208/31,9208 7[] 100,00% 1,9568/s 31,9208 217ms5tart at 2022 |
| 11-02 06:16:33.630703553 +0000 UTC |
| |
| |
| Parallel: 5 Jobs: 5 |
| Threshold: 50.00MB PartSize: auto |
| VerifyLength: false VerifyMd5: false |
| CheckpointDir: /root/.obsutil_checkpoint |
| |
| |
| Upload successfully, 31.92MB, n/a. /root/.jenkins/workspace/test-demo/target/cas-demo-1.0-SMAFSHOT.jar> obs://test-hzp-123/cas-demo-1.0-SMAFSHOT.jar, cost [216], status [200], request id |
| [0000014436FBF8664011C5BD4CFFFFA0] |
| upgrade component |
| |
|]00_1d 4d111d3-1a91-4300-b230-21094b591162 |
| 2 |
| 100_10_ 4011(022-1881-4200-0220-21084004(105 |
| Dentied 1 |
| (ripeling) // stars |
| (ripering // stage |
| (rapeller) / rode |
| Pineline End of Pineline |
| Finished: SUCCESS |
| |
| |
| |

- 步骤6 登录CAE控制台。
- 步骤7 导航栏单击"组件列表",查看对应组件的版本号已更新,如图3-33。

图 3-33 查看组件版本号

| 🕼 环境: test | • 🕀 | 🔞 应用: hzp | * | | | | | | | |
|--------------------------|-----|----------------------------------|-------|--------------|----------------|--------|-------------------------------|-----------------|------------|-----|
| 新聞組件 | | | | | | | | | 调输入事件套进行搜索 | Q C |
| 名称重本 1 | = | 代码源 🏹 | ne V | 实制个数 (可用/全部) | 实制现储 | 外部站间地址 | 修葺时间 15 | 快速影响 | 操作 | |
| Janen test-jær v1.0.3 | | 校注句 cae-demo-1.0-SNAPSHOT.jar | ◎ 运行中 | 1/1 | 0.5Core 1GiB | | 2022/11/01 16:16:39 GMT+08:00 | 1 0 1 1 1 | 升级 更多▼ | |
| | | | | | | | | | | |

----结束

3.4.2 Gitlab 自动触发 Jenkins 构建

Gitlab触发Jenkins构建有以下两种方式:

- 通过配置好的Webhook来Push events,触发Jenkins构建任务。
- 修改构建配置指定分支的文件来Push events,触发Jenkins构建任务。

示例通过方式一来触发Jenkins构建。

步骤1 登录Gitlab,进入代码仓库,示例为"test-demo",单击"Settings",选择 "Webhooks",在右下角的"Test"下拉框,选择"Push events",如<mark>图3-34</mark>所 示。

- 🖊 GitLab 🛛 = Menu • mo Push events O Project inform Branch name or wildcard pattern to trigger on (leave blank for all) URL is triggered by a push to the repository Repository Dr Issues Tag push events
 URL is triggered when a new tag is pushed to the repository Comments
 URL is triggered when someone adds a comment 🕼 Cựcd COCO
 Security & Complian
 Deployments
 Monitor
 Infrastructure Confidential comments
 URL is triggered when someone adds a comment on a confidential issue URL is triggered when an issue is created, updated, closed, or reopened 🖰 Packages & Registries Confidential issues events
 URL is triggered when a confidential issue is created, updated. Merge request events
 URL is triggered when a merge request is created, updated, or merged Snippets
 Settings
 General Job events
 URL is triggered when the job status changes General Integrations Webhooks Access Tokens Repository CVCD Monitor Pipeline events
 URL is triggered when the pipeline status changes Wiki page events
 URL is triggered when a wiki page is created or updated Deployment events URL is triggered when a deployment starts, finishes, fails, or is canceled URL is triggered when a feature flag is URL is triggered when a feature flag is URL is triggered when a release is created by the sents URL is triggered when a release is created by the sents SSL verification Confidential issues events Confidential issues events
 Note events
 Confidential note events
 Merge requests events
 Job events Add webhook Project Hooks (1) Pipeline events Test ~ Edit Delete
- 图 3-34 Webhook 触发 Jenkins 构建

步骤2 登录Jenkins, 左侧构建执行状态中可以看到已经触发的构建任务,单击构建任务编号,选择"Console Output",查看构建输出日志,如图3-35。

图 3-35 查看构建任务状态

| 🏘 Jenkins | | | | | | Q 查1 | Ŷ | 0 | admin 🛨 🕄 |
|--------------|--|---------|---------------|----------|----------------|------|----------------|---------------|----------------------|
| Dashboard > | | | | | | | | | |
| 쯜 新建任务 | | | | | | | | | / 15 10 10 10 |
| 鵗 用户列表 | | REAL | + | | | | | | |
| 📄 构建历史 | | | | | | | | | |
| 🔍 项目关系 | | 5 | W 名称: | 上次成功 | 上次失败 | | 上次持续时间 | | |
| 🖭 检查文件指纹 | | \odot | iội test-demo | 12 分 #75 | 1 (사람 3 分) 🔒 👫 | | 1分22秒 | | Þ |
| 🏠 系統管理 | | | | | | | | | |
| 🌯 FAESHAR | | 副示 小 | Φ 大 | | | 1944 | ふ Atom feed 全部 | Atom feed 938 | Atom feed \$84819462 |
| 新建规图 | | | | | | | | | |
| 构建队列 | ^ | | | | | | | | |
| 队列中空和构建任务 | | | | | | | | | |
| 构建执行状态 | ^ | | | | | | | | |
| 1 596 | | | | | | | | | |
| 2 test-siemo | #76 * | | | | | | | | |
| | 2 2 2 2 5 2 2 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 | | | | | | | | |
| | Rest Consider Conside | e. | | | | | | | |
| | 📋 \$2888 | | | | | | | | |
| | 🚸 Git Build Da | 50 | | | | | | | |

步骤3 查看构建输出日志,构建出来的是jar包,部署成功,如图3-36。

图 3-36 查看构建输出日志

| parameters 1 "hose inser": "100 215-20202/on mer conformal@e_larghl-1.0.6" |
|---|
| Dasa_Intage : 100. 210.2000/09_300_020/04130K0~141001.1.0.0 |
| |
| In The Association of the Associ |
| resource_inart i |
| cpulinat count, |
| nemory_limit : 101 |
| b to see the total sector to the total sector total secto |
| acces_into : U. |
| hage_uii: swt.crrmoithr/.mynusmeicloud.com/pas_cse /test-jar:vo, |
| available_replica : 1, " |
| juujii aiiiiisa aa a |
| oulud_lu _ yokogudekine/itselkak/caslulz/oakodewkigenS . *tatato: Turunia/ |
| status rumning, |
| pulld_log_ld : |
| |
| |
| cuitent version: 1.0.3 |
| margina instaine: 1.0.4 |
| appland versions. I so the |
| ариона јак со нов Генера — 1 д. 8766 2/е 1 КОНВУ/11 02009 21 |
| |
| 06:34:25.436303076 +0000 UTC |
| |
| |
| Parallel: 5 Jobs: 5 |
| Threshold: 50.00MB PartSize: auto |
| VerifyLength: false VerifyMd5: false |
| CheckpointDir: /root/.obsutil_checkpoint |
| |
| |
| Upload successfully, 31.92MB, n/a, /root/.jenkins/workspace/test-demo/target/cae-demo-1.0-SMAPSHOT, jar> obs://test-hzp-123/cae-demo-1.0-SMAPSHOT.jar, cost [621], status [200], request id |
| [00000184370C5324401440C42F2467DA] |
| upgrade component |
| |
| "job_id": "8s4baf15-8073-44ca-ade0-1a9f6ebt5180" |
| |
| job_id= 8a4baf15-8073-44carade0-1a9f6ebf518b |
| |
| [Pipeline] } |
| [Pipeline] // stage |
| (Pipeline) |
| [Pipeline] // node |
| [Pipeline] End of Pipeline |
| Findhad SIDTER badbing |
| |
| |

- **步骤4**登录CAE控制台。
- 步骤5 左侧导航栏单击"组件列表",查看对应组件的版本号已更新,如图3-37。

图 3-37 查看组件版本号

| \$ 环境: 🤒 test | • 🕀 | 📦 应用: app | · · | Ð | | | | | | |
|-------------------------------------|--------|-----------------------|-------|---|--------------|------|---|-------------------------------|-----------|---------|
| | | | | | | | | | | |
| 新增组件 | | | | | | | | | 请输入名称进行搜索 | Q C 🕲 |
| 名称/版本/ID ≑ | f | (明源 7 | 秋恋 🏹 | | 实例个数 (可用/全部) | 访问地址 | 最近一次変更状态时间 ≑ | 创建时间 💠 | 快速跳转 | 操作 |
| 1est-521 v1.0.0 47df1d0e-d4c9 | ទ n | ໝ ginx stable-perl | 3 运行中 | | 1/1 | - | ● 部署成功 Q 2024/05/21 23:14:56 GMT+08:00 | 2024/05/21 23:14:54 GMT+08:00 | r 9 r 8 | 升级 更多 ▼ |
| | | | | | | | | | | |

----结束

3.5 附录

3.5.1 AK/SK 获取方法

- 步骤1 使用管理员账号登录CAE控制台。
- 步骤2 单击右上角的用户名,在下拉菜单选择"我的凭证"。
- 步骤3 在导航栏左侧选择"访问密钥"。
- 步骤4 单击"新增访问密钥",通过身份认证后成功创建AK/SK。
- 步骤5 单击"立即下载"。
- 步骤6 下载成功后,在credentials文件中获取AK和SK信息:
 - Access Key Id的值即为AK。
 - Secret Access Key的值即为SK。

注意事项:

- 1. 每个用户仅允许保留2个有效的访问密钥。
- 为保证访问密钥的安全,访问密钥仅在初次生成时自动下载,后续不可再次通过 控制台界面获取,请妥善保管访问密钥。

----结束

4 Jenkins 流水线支持多组件按照依赖顺序部

4.1 概述

适用场景

使用jenkins来构建部署升级微服务组件,升级涉及多个微服务组件,每个组件有单独的流水线用于构建部署升级。但是组件之间存在依赖关系,必须等待所依赖的组件升级完成才能升级流水线,否则会有基本功能问题。

升级过程需要时刻关注所依赖的组件是否升级完成,才能进行下一个组件的升级任务。

解决方案

新建一条jenkins流水线来编排多个组件,自动按照依赖顺序构建升级,并使用参数化 构建的方式,同时支持单个或多个组件的升级。

下面以一个示例说明流水线执行流程。

有4个组件分别是componentA、componentB、componentC和componentD,其中 componentD依赖componentB和componentC,componentB和componentC依赖 componentA。

流水线执行流程:



4.2 前提条件

组件要求

每个待升级的组件已完成配置流水线构建升级部署CAE。

配置应用无损上线

升级多个组件时,需确保上一个组件已经就绪,升级才返回成功,再升级下一个组 件。因此,您需要配置应用无损上线,具体操作请参考健康检查的<mark>使用就绪探针保证</mark> <mark>升级时流量正常</mark>章节进行配置。

若您的组件是SpringCloud应用,则请参考Spring Cloud应用无损上线。

安装 Jenkins 插件

| 步骤1 | 进入 | jenkins的系统管理页面, | 単击 | "插件管理" | 0 |
|-----|----|-----------------|----|--------|---|
|-----|----|-----------------|----|--------|---|

图 4-1 插件管理

| 🏘 Jenkins | | | c | 入 直找 |
|----------------------|--|--|----------------------------|------|
| Dashboard | | | | |
| 🝧 新建任务 | 管理Jenkins | | | |
| 鵗 用户列表 | | | | |
| 🔁 构建历史 | Building on the built-in node can be a security issue. You should se | et up distributed builds. See the documentation. | | |
| 🔍 项目关系 | | | | |
| 🚛 检查文件指纹 | 系统配置 | | | |
| 🔅 系统管理 | 系统配置 0 | 会局工具配置 | 上 插件管理 | |
| 🍓 我的视图 | 配置全局设置和路径 | / 工具配置,包括它们的位置和自动安装器 | 添加、删除、禁用或启用Jenkins功能扩展活件 | ÷. |
| 👒 Lockable Resources | | | | |
| 🛅 新建视图 | 安全 | | | |
| 构建队列 | ▲局安全配置 Jenkins 安全,定义谁可以访问或使用系统。 | Manage Credentials Configure credentials | 凭据配置 配置凭据的提供者和类型 | |
| 队列中没有构建任务 | | | | |

步骤2 在"插件管理"页面,单击"可选插件",搜索"pipeline"和"extended choice parameter"两个插件进行安装,安装时选择不重启安装。

| =台 kins | Plugin Manager जहास जित्तालन हरू हरू | Q pipeline |
|------------|--|-------------|
| | Irstall Name 1 | Released |
| | Pipeline 2.5 Pipeline 2.5 Pipeline 2.5 Pipeline 2.5 Pipeline 3.5 Pi | 5年0月 ago |
| | Pipeline: Github Groovy Liberries 10 pipeline: github Allows Pipeline: Github. Allows Pipeline: Github. | 6年9月 ago |
| | Ducker Pipeline 115 Ducker Pipeline 126 professe THZ Socker Build and use Ducker containers from pipelines. | 2 年7 月 ago |
| | Pipeline: Declarative Agent API 111 pipeline: @dl pipeline: @dl pipeline: @dl pipeline: Declarative Extension Pavints API plagin. This plagin is deprecated. In general, this means that it is either elasolete, no longer being developed, or may no longer work. Learn more. | 6 年 7 月 ago |
| | Pipeline SCM API for Blue Ocean 1246 This plogin is a part of BlueOcean Flugin | 2年6月ago |
| | Pipeline implementation for Blue Ocean 124.6 Pipeline (MPPR) The plugin is a part of Blue/Ocean Riugin The plugin is a part of Blue/Ocean Riugin | 2 年 6 月 ago |

图 4-2 安装 pipeline 插件

图 4-3 安装 extended choice parameter 插件

| Dashboard , 插件管理 | | |
|---------------------|---|-----------------------------|
| | Plugin Manager সর্ল <mark>যর্জন</mark> তৎ রঞ | Q extended choice parameter |
| | Install Name 1 | Released |
| | Extended Choice Parameter 0.82 Contraction: Adds extended functionality to Choice parameter | 3 年 1 月 ago |
| | Optimite Estimated Colorize Planameter 10.1 The plugin is a sample plugin to explain how to write a Hudson plugin. Warning: This plugin version my on the safe to use. Please review the following security notices: • Basenond stored in stain.text | 9年6月 ago |
| | Install without restart Download now and install after restart 0 小时 42 分 之防行机了使指行指数. 33期间数 | |

----结束

4.3 操作步骤

场景说明

当前示例包含4个组件,分别为componentA、componentB、componentC和 componentD。

组件依赖关系为: componentD依赖componentB和componentC, componentB和 componentC依赖componentA。

现在使用jenkins流水线来编排部署这4个组件,支持单个或多个组件升级而不用考虑依赖关系单个组件依次升级。

配置 jenkins 流水线

步骤1 进入jenkins的系统管理页面,单击"新建任务"。

图 4-4 新建任务

| 🏘 Jenkins | | | |
|----------------------|------------------|--------------|--------------|
| Dashboard + | | | |
| 🖀 新建任务 | | | |
| 鶡 用户列表 | cae-backend 所有 + | | |
| 🔁 构建历史 | | | |
| 🔍 项目关系 | S W 名称⊥ | 上次成功 | 上次失败 |
| ▲ 检查文件指纹 | 📀 🛆 cae-backend | 1天8小时 #49 | 1天11小时 #48 |
| 🔆 系統管理 | ⊘ 🖄 componentA | 2 小时 59 分 #7 | 7 小时 14 分 #5 |
| 🚷 我的视图 | ✓ XQX componentB | 2 小时 58 分 #7 | 无 |
| 👒 Lockable Resources | ComponentC | 2 小时 58 分 #7 | 1天5小时 #5 |
| 新建视图 | ComponentD | 2 小时 57 分 #5 | 无 |

步骤2 输入任务名称,选择任务类型为"流水线"。

步骤3 在"General"页面中,添加"扩展选项参数"。

图 4-6 添加参数

| 🏘 Jenkins | |
|--------------------------------|---|
| Dashboard > pipeline-arrange > | |
| | |
| G | neral 构建触发器 高级项目选项 流水线 |
| 描述 | |
| | |
| | |
| | |
| | |
| [約4] | 7本) 预览 |
| 0 | GitHub 顶目 |
| Gitl | ab Connection |
| ai | lah |
| | 100 |
| | Use alternative credential |
| | Preserve stashes from completed builds ? |
| | Track Git source of another project |
| | 不允许并发构建 |
| | 丢弃旧的构建 ? |
| | |
| 1. | |
| | Extended Choice Parameter |
| | |
| | 子付参数 <u>工</u> , 本工 年 勤 |
| 构 | 本の信参数 |
| | |
| | 文本参数 |
| | 运行时参数 d to GitLab. GitLab webhook URL: http://100.95.152.43:8080/project/pipeline-arrange ? |
| | 选项参数 |
| | GitHub hook trigger for GITScm polling ? |

步骤4 参考图4-7配置需要编排的组件。

| 参数名称 | 描述 | 示例 |
|----------------------------|-------------------------------|---|
| Name | 设置参数的名称,构建时选择的组件会存放在 该变量名。 | Model_Name |
| Description | 设置构建参数的描述。 | 选择需要升级的组 件 |
| Parameter Type | 选择基本参数类型。 | Check Boxes |
| Number of Visible Items | 设置流水线编排的组件数量。 | 4 |
| Delimiter | 设置组件之间的分隔符,设置为逗号(,)。 | , |
| Value | 设置流水线编排的组件名称,用逗号(,)分 隔。 | componentA,com ponentB,compon entC,component D |

表 4-1 配置构建参数

| 图 4-7 | 配置构建参数 |
|-------|--------|
|-------|--------|

Dashboard • pipeline-arrange

| Extended Choice Para Name ? | neter 构建财选择的组件合 | |
|--------------------------------|--------------------|------------------|
| | 存放在这个变量名 | |
| Model_Name | | |
| Description | 构建时选择参数的描述 | |
| Description | | |
| 选择需要升级的组件 | | |
| | - 法将基本参数类型 | |
| Parameter Type ? | es Alter Parks | |
| Chark Boyer | 选择Check Boxes | |
| CHECK DOAES | | |
| Number of Visible | items 流水线编排的组件数量 | |
| G. | | |
| Ľ | | |
| Delimiter ? | 组件之间的分隔符,英文逗号 | |
| | | |
| Ľ | | |
| Quote Value | | |
| | | 7 |
| Choose Sourc | e for Value | 流水线编排的组件名称,用逗号分隔 |
| Value | | |
| Value | | |
| | | |

步骤5 选择"流水线",在"流水线"页面进行脚本配置,脚本内容请参考流水线脚本。

图 4-8 配置流水线脚本

Dashboard

| <pre>Performance Action and Actio</pre> | |
|---|--|
| <pre>Product State ##** ********************************</pre> | |
| <pre>W * *</pre> | Puterine Skripk |
| <pre>1 * pistine { 1 * pistine</pre> | 即本 ? |
| 29 //不知faifattrum, jumailaip未能的建築者。如果其他并行的taugh形式。直到简者的并行在希知行其成 //faifattrum //指制的app. 30 //mailat app. 31 //mailat app. 32 //mailat app. 33 //mailat app. 34 //mailat app. 35 //mailat app. 36 //mailat app. 36 //mailat app. 37 //mailat app. 38 //mailat app. 39 //mailat app. 30 //mailat app. 31 //mailat app. 31 //mailat app. 31 //mailat app. 32 //mailat app. 32 //mailat app. 32 //mailat app. 33 //mailat app. 34 //mailat app. | <pre>i tprile() territor() t</pre> |
| 12 ///warlads 本売豊富的にapp急性作用行用、2算其中有一个stapp没计关系。例*mild paralled jobs*位个stapp没未有的,它面接的stapp不会执行 parallad (lic concentry) 13 (main functionement) 14 (main functionement) 15 (main functionement) 16 (main functionement) 17 (main functionement) 18 (main functionement) 19 (main functionement) 10 (main functionement) 10 (main functionement) 10 (main functionement) 13 (main functionement) 14 (main functionement) 15 (main functionement) 16 (main functionement) 17 (main functionement) 18 (main functionement) 19 (main functionement) | //不加failsatture monaller/10050mger开发前一下大张,其UH/111分位由于正 //不加failsatture, parallel/天校的stage不会影响其他并行的stage队行,直到所有的并行任务执行完成 //failFast true //搜索参加 |
| 34* tige("Eull a component") { 35 when { endrowneth maxe: 'componentB', value: 'componentB') } 36* tige { 37 th ' exho 'tart to build componentB''''' 38 context to build componentB''''''''''''''''''''''''''''''''''' | 32 //parallel 表示理面的stage是并行执行的,只要其中有一个stage执行失败,则"Build parallel jobs"这个stage是失败的,它后结约stage不会执行 33 - parallel (|
| 37 ah "" endo "start to build component?" "" 38 + soriar(39 def componentBuild=build(bit) "component?") 40 metrinth commonship (did setMetrin(1)) | <pre>stage(sull components) { 35</pre> |
| | 37 sh.'' edo 'start to build component8'''' 38 + script(script(39 (efc component8uld+build(shc)''component8') 40 mort this component16 (effectual()) |
| | ☑ 使用 Groovy 沙盒 ? |

步骤6 脚本配置完成后,单击"保存"。

----结束

流水线脚本

流水线脚本支持componentA、componentB、componentC和componentD,4个组件中的一个或多个同时升级,并且按照依赖顺序先升级componentA,再并行升级 componentB和componentC,最后升级componentD。如果构建时没有勾选到的组件 会跳过,不执行升级。

pipeline { agent any

```
environment{
    //构建时选择的组件以','隔开的格式存在变量Model_Name里
    //判断变量Model_Name里的可选组件是否存在,如果存在,把组件名赋值给对应变量,如果不存在,把字
符串"error"赋值给对应变量。
    //用于判断后续组件是否需要构建
    _componentA="${sh(script:' echo $Model_Name| grep -w -o "componentA" || echo "error" ',
returnStdout: true).trim()}"
    _componentB="${sh(script:' echo $Model_Name| grep -w -o "componentB" || echo "error" ',
returnStdout: true).trim()}"
    _componentC="${sh(script:' echo $Model_Name| grep -w -o "componentC" || echo "error" ',
returnStdout: true).trim()}"
     _componentD="${sh(script:' echo $Model_Name| grep -w -o "componentD" || echo "error" ',
returnStdout: true).trim()}"
  }
  stages {
    stage('Build componentA') {
       //when条件判断当环境变量"_componentA" = "componentA"时,表示构建时选择了该组件,执行此
stage,否则跳过此stage
      when { environment name: '_componentA', value: 'componentA' }
      steps {
    sh "' echo "start to build componentA" '"
        script{
          //build(job: 'componentA') 表示执行componentA任务,这个任务是jenkins中创建的任务名称,用
于componentA组件升级的任务
          def componentBuild=build(job: 'componentA')
          //打印执行任务的结果
          println componentBuild.getResult()
        }
      }
    }
    stage('Build parallel jobs') {
      //failFast true 表示parallel并行的stage中只要有一个失败,其他并行任务也会中止
      //不加failFast true, parallel中失败的stage不会影响其他并行的stage执行,直到所有的并行任务执行完
成
       //failFast true //按需添加
       //parallel 表示里面的stage是并行执行的,只要其中有一个stage执行失败,则"Build parallel jobs"这个
stage是失败的,它后续的stage不会执行
      parallel {
         stage('Build componentB') {
           when { environment name: '_componentB', value: 'componentB' }
           steps {
             sh " echo "start to build componentB" "
             script{
               def componentBuild=build(job: 'componentB')
               println componentBuild.getResult()
             }
           }
         }
         stage('Build componentC') {
           when { environment name: '_componentC', value: 'componentC' }
           steps {
             sh " echo "start to build componentC" "
             script{
               def componentBuild=build(job: 'componentC')
               println componentBuild.getResult()
             }
           }
         }
      }
    stage('Build componentD') {
      when { environment name: '_componentD', value: 'componentD' }
      steps {
        sh " echo "start to build componentD ""
        script{
          def componentBuild=build(job: 'componentD')
          println componentBuild.getResult()
```

} } }

}

脚本说明

每个组件都需要在pipeline中的envionment获取组件参数,用于判断参数化构建时是 否选中,执行下面脚本:

_{component_name}="\${sh(script:' echo \$Model_Name| grep -w -o "{component_name}" || echo "error" ', returnStdout: true).trim()}"

表 4-2 参数说明

| 参数名称 | 描述 |
|------------------|---|
| Model_Name | 配置构建参数中的Name。 |
| {component_name} | 配置构建参数中的Choose Source for Value中的value里面填写的组件名称之 一。 |

每个组件都需要配置一个stage,用于组件的升级任务,如果是串行执行的,则按照先 后顺序放置stage的位置,如果是并行执行的任务就放置在stage('Build parallel jobs') 中的parallel中。

stage脚本:

```
stage('Build {component_name}') {
```

```
when { environment name: '_{component_name}', value: '{component_name}' }
steps {
    sh "" echo "start to build {component_name}" ""
    script{
        def componentBuild=build(job: '{component_jenkins_task}')
        //打印执行任务的结果
        println componentBuild.getResult()
    }
}
```



}

| 参数 | 描述 |
|--------------------------|---|
| {component_name} | 配置构建参数中的Choose Source for Value中的value里面填写的组件名称之 一。 |
| {component_jenkins_task} | jenkins中创建的用于组件升级的任务名 称,例如:用于componentA组件升级的 任务名称为如 <mark>图4-9</mark> 所示。 |

图 4-9 组件升级任务名称

| 🏟 Jenkins | | | | | | | Q 直找 | 0 | 9 1 | 💄 admin | 日注機 |
|--------------------------|------|---------|-----|-------------|-------------|-----------|------|--------|-----|---------|----------|
| Dashboard , admin , 我的视图 | ● 所有 | ī • | | | | | | | | | |
| 🍧 新建任务 | | | | | | | | | | | 15403340 |
| 🍓 用户列表 | | RTAT | + | | | | | | | | |
| 🔁 构建历史 | | | | | | | | | | | |
| 🔍 项目关系 | | s | w | 名称: | 上次成功 | 上次失败 | | 上次持续时间 | | | |
| 检查文件指纹 | | \odot | ۵ | cae-backend | 7天1小时 #49 | 7天4小时 #48 | | 31 89 | | ⊳ | |
| | | \odot | IỘI | componentA | 23 小편 #14 | 4天20小时 #8 | | 32 H | | ⊳ | |
| 构酿队列 | ^ | \odot | :ộ: | componentB | 23 Jv8j #13 | 无 | | 1分29秒 | | ⊳ | |
| 队列中设有构建任务 | | \odot | :ộ: | componentC | 23 小时 #13 | 6天22小时 #5 | | 45 秒 | | ⊳ | |

使用流水线升级多个组件

步骤1 选择流水线,单击"参数化构建"。

| 图 4-10 参数化 | 幼建 | | | | |
|----------------------|----|------------|---------------------------------|--------------|------------|
| 🏘 Jenkins | | | | | |
| Dashboard | | | | | |
| 쯜 新建任务 | | | | | |
| 鵗 用户列表 | | cae-backen | d 所有 + | | |
| 🔁 构建历史 | | | | | |
| 🔍 项目关系 | | s v | V 名称↓ | 上次成功 | 上次失败 |
| 🚛 检查文件指纹 | | 0 | Cae-backend | 2 天 5 小时 #49 | 2天8小时 #48 |
| 🏠 系統管理 | | ં | componentA | 23 小时 #7 | 1天4小时 #5 |
| 鵗 我的视图 | | Ø 1 | componentB | 23 小时 #7 | 无 |
| 👒 Lockable Resources | | \odot | > componentC | 23 小时 #7 | 2天2小时 #5 |
| 新建规图 | | 0 x | componentD | 23 小时 #5 | 无 |
| 构建队列 | ^ | 0 | ▶ pipeline-arrange ▼ | 23 小时 #21 | 1天4小时 #19 |
| 队列中没有构建任务 | | | 交更历史 S Build with Parameters | 没有 | 无 |
| 构建执行状态 | ^ | 0 x | | 3月17天 #144 | 3月17天 #136 |
| 1 空闲 2 空闲 | | 💮 X | | 没有 | 无 |
| | | 💮 X | | 没有 | 无 |

步骤2 选择要升级的组件,可以选择一个或多个同时升级,本示例选择全部组件同时升级,单击"开始构建"。

图 4-11 选择组件

| 🧌 Jenkins | |
|--|--|
| Dashboard > pipeline-arrange > | |
| ▲ 返回工作台 気 状态 ご 変更历史 | Pipeline pipeline-arrange 需要如下参数用于构建项目: |
| Build with Parameters | Model_Name ComponentA 送择需要升级的组件 |
| 촳 配置 🚫 删除 Pipeline | componentC componentD |
| 》 移动 | 开始构建 |
| 完整阶段视图 重命名 | |
| ? 流水线语法 | |

步骤3 可在构建历史中查看流水线任务执行结果。

图 4-12 查看执行结果

| 🧛 Jenkins | | | | | | |
|---|---|-------------------|----------------|------------------------------------|------------------------------------|--------------------------|
| Dashboard > pipeline-arrange > | | | | | | |
| ★ 返回工作台 | Pineline ninelin | a-arrang | 9 | | | |
| 🔍 状态 | i ipenne pipenne | e-arrange | 5 | | | |
| 📄 变更历史 | | | | | | |
| Suild with Parameters | 40033400 | | | | | |
| 🐡 配置 | 最近变更 | | | | | |
| 🚫 删除 Pipeline | 阶段视图 | | | | | |
| 1 移动 | | Build | Build parallel | Build | Build | Build |
| 🔍 完整阶段视图 | | componentA | jobs | componentB | componentC | componentD |
| 🔁 重命名 | Average stage times: | 53s | 62,000 | | | |
| | (Average <u>full</u> run time: ~4min 27s) | | 02IIIS | 44s | 1min 44s | 29s |
| 2 流水线语法 | (Average <u>full</u> run time: ~4min 27s) | 35s | 53ms | 44s 36s | 1min 44s | 29s 41s |
| itx水线语法 ity 相違历史 へ | (Average full run time: ~4min 27s) | 35s | 53ms | 36s | 1min 44s | 29s 41s |
| | (Average <u>full</u> run time: ~4min 27s) | 35s 55s | 53ms | 44s 36s 1min 21s | 1min 44s 1min 21s | 295 415 515 |
| ● 流水线菌法 ◇ 読水线菌法 ◇ Build History 构建历史 へ ○ Filter builds ○ #22 2023年10月20日下午6月8 ~ ○ #25 ② 安更历史 ○ #25 ③ Console Curput | (Average full run time: ~4min 27s) | 35s 55s 35s | 53ms | 44s 36s 1min 21s 1min 20s | 1min 44s 1min 21s 37s 50s | 295 415 515 395 |

步骤4 流水线日志如下所示,可以看出先升级componentA,再并行升级componentB和 componentC,最后升级componentD。

Started by user admin Running in Durability level: MAX_SURVIVABILITY [Pipeline] Start of Pipeline [Pipeline] node Running on Jenkins in /root/.jenkins/workspace/pipeline-arrange [Pipeline] { [Pipeline] sh + echo componentA,componentB,componentC,componentD + grep -w -o componentA [Pipeline] sh + echo componentA,componentB,componentC,componentD + grep -w -o componentD [Pipeline] sh + echo componentA,componentB,componentC,componentD + grep -w -o componentB [Pipeline] sh + echo componentA,componentB,componentC,componentD + grep -w -o componentC [Pipeline] withEnv [Pipeline] { [Pipeline] stage [Pipeline] { (Build componentA) [Pipeline] sh + echo 'start to build componentA' start to build componentA [Pipeline] script [Pipeline] { [Pipeline] build Scheduling project: componentA Starting building: componentA #13 [Pipeline] echo SUCCESS [Pipeline] } [Pipeline] // script [Pipeline] } [Pipeline] // stage [Pipeline] stage [Pipeline] { (Build parallel jobs) [Pipeline] parallel [Pipeline] { (Branch: Build componentB) [Pipeline] { (Branch: Build componentC) [Pipeline] stage [Pipeline] { (Build componentB) [Pipeline] stage [Pipeline] { (Build componentC) [Pipeline] sh [Pipeline] sh + echo 'start to build componentB' start to build componentB [Pipeline] script + echo 'start to build componentC' start to build componentC [Pipeline] { [Pipeline] script [Pipeline] { [Pipeline] build Scheduling project: componentB [Pipeline] build Scheduling project: componentC Starting building: componentB #12 Starting building: componentC #12 [Pipeline] echo SUCCESS [Pipeline] } [Pipeline] // script [Pipeline] } [Pipeline] // stage [Pipeline] } [Pipeline] echo SUCCESS [Pipeline] } [Pipeline] // script [Pipeline] } [Pipeline] // stage [Pipeline] } [Pipeline] // parallel [Pipeline] }

[Pipeline] // stage [Pipeline] stage [Pipeline] { (Build componentD) [Pipeline] sh + echo 'start to build componentD' start to build componentD [Pipeline] script [Pipeline] script [Pipeline] build Scheduling project: componentD #10 [Pipeline] building: componentD #10 [Pipeline] echo SUCCESS [Pipeline] cho SUCCESS [Pipeline] } [Pipeline] // script [Pipeline] } [Pipeline] // stage [Pipeline] } [Pipeline] // withEnv [Pipeline] } [Pipeline] // node [Pipeline] End of Pipeline Finished: SUCCESS

步骤5 登录CAE控制台,在组件列表查看组件状态,"最近一次变更状态/时间"列显示状态为"升级成功",表示组件升级成功。

图 4-13 查看组件

| 概范 ···································· | 新潮油并 | | | | | | | 请输入名称进行接来 | QC |
|--|---------------------------------------|---------------------|-------|--------------|------|--------------------------|-------------------------------|-----------|--------|
| 实例列表 | 名称/版本/ID ⇔ | 代码源 🍞 | 秋志 🏹 | 实例个数 (可用/全部) | 访问地址 | 最近一次变更状态时间 ≑ | 创建时间 🖕 | 快捷跳转 | 操作 |
| 组件配置 组件事件 | mginx-0522 v1.0.0 ead9016F-88a8 | tõ€®n nginx perl | 3 运行中 | 1/1 | - | 升級成功 | 2024/05/22 22:47:41 GMT+08:00 | 18 e 18 e | 升级 更多▼ |

----结束

5 SpringCloud 应用部署到 CAE 自动对接 Nacos 引擎

5.1 概述

应用场景

用户可以将业务代码打包部署到CAE,只需在部署的组件配置中添加Nacos引擎配置, 即可用自动化对接Nacos引擎,无需手动修改业务代码配置文件,避免人为误操作。

本章节通过CAE源码部署能力自动化部署一个provider服务和一个consumer服务,帮助您体验自动接入Nacos引擎。

流程优化对比

当前对接Nacos流程:



原理说明

在Spring Cloud应用中,系统环境变量的优先级高于配置文件。因此,如果在系统环 境变量中配置了参数,那么系统会优先使用环境变量中的参数。为了实现自动对接 Nacos引擎,CAE会自动在组件中添加 SPRING_CLOUD_NACOS_DISCOVERY_SERVERADDR和 SPRING_CLOUD_NACOS_CONFIG_SERVERADDR环境变量。这样,即使在配置文件 中没有配置这些参数,系统也能够正确地连接到Nacos引擎。

配置nacos参数时需注意:如果用户在外部配置了比环境变量优先级更高的nacos注册 发现参数,可能会导致nacos对接异常。例如,如果在启动命令中添加了-spring.cloud.nacos.discovery.server-addr=nacos.com,这会覆盖环境变量中的相应配 置,从而导致组件对接nacos异常无法正常工作。因此,在配置nacos参数时,需要注 意优先级的问题,以确保nacos能够正确地对接。

5.2 前提条件

- 已创建环境和应用。
- 已创建Nacos引擎,具体操作请参考创建Nacos引擎。
 Nacos引擎所在虚拟私有云和子网同CAE环境的VPC一致。
- 登录GitHub,将Nacos源码fork到您的仓库下。源码地址:https://github.com/ nacos-group/nacos-examples。

5.3 操作步骤

源码仓库授权

- 步骤1 登录CAE控制台。
- 步骤2选择"系统设置"。
- **步骤3** 在"系统设置"页面,单击"源码仓库授权"模块的"编辑",进入"已授权源码仓库"页面。
- 步骤4 单击"新建授权",进入"新建授权"页面。
 - 1. 单击"GitHub"。
 - 2. 授权名称输入: nacos-github。
 - 3. 授权方式选择"OAuth"。
 - 4. 单击"使用OAuth授权"。
- **步骤5** 在"服务声明"弹框中,勾选"我已知晓本服务的源码构建功能收集上述信息,并同意授权对其的收集、使用行为。"。单击"确认"完成授权。
- 步骤6 授权完成后,进入GitHub登录页面。
- **步骤7** 输入用户GitHub的用户名或邮箱和密码,单击"Sign in",等待OAuth认证完成,返回CAE页面。
- 步骤8 在CAE "系统设置"页面提示框单击 "确认",完成源码仓库授权。

----结束

创建并部署 nacos-provider 组件

- 步骤1 在导航栏中选择"组件列表"。
- 步骤2 在页面上方,下拉选择<mark>前提条件</mark>中已创建的应用和环境,单击"新增组件"。

步骤3 参考表5-1配置组件信息。

| 参数 | 说明 |
|------------------|--|
| 组件名称 | 新建组件的名称。本实践输入名称为"nacos-provider"。 |
| 版本号 | 组件的版本号。 本实践版本号为1.0.0。 |
| 实例规格 | 选择实例规格,例如:0.5core、1GiB。 |
| 实例数量 | 输入实例数为1。 |
| 代码源 | 选择"源码仓库 > GitHub",配置授权信息。 • 授权信息: nacos-github。 • 用户名/组织: 用户授权使用的用户名/邮箱。 • 仓库名称: nacos-example。 • 分支: master。 |
| 语言/运行时 | 选择"Java17"。 |
| 自定义构建 | 选择"使用自定义命令"。 小黑框内输入: cd ./nacos-spring-cloud-example/nacos-spring-cloud-discovery-example/nacos- spring-cloud-provider-example && mvn clean package |
| Dockerfile地 址 | 输入"./"。 |

表 5-1 nacos-provider 组件基本信息

- 步骤4 单击"配置组件"。
- 步骤5 在"组件配置"页面,单击"微服务引擎CSE"模块的"配置",进入"微服务引擎 CSE配置"页面。
- 步骤6 单击"注册配置中心",选择前提条件中您已创建的Nacos引擎。
- 步骤7 单击"保存",完成对该组件微服务引擎CSE配置。
- **步骤8** 在"组件配置"页面,单击"环境变量"模块的"编辑",对nacos-provider组件进行 配置。
- 步骤9 单击"新增环境变量",输入变量名称及变量。
 - 变量名称: JAVA_TOOL_OPTIONS。
 - 变量/变量引用: --add-opens=java.base/java.lang=ALL-UNNAMED --addopens=java.base/java.lang.reflect=ALL-UNNAMED。
- 步骤10 单击"确定",完成对该组件环境变量配置。
- 步骤11 在"组件配置"页面,单击"配置并部署组件"。
- 步骤12 确认配置信息无误后,单击"确定",等待组件部署完成。

----结束

创建并部署 nacos-consumer 组件

- 步骤1 在导航栏中选择"组件列表",返回组件列表页面。
- 步骤2 在页面上方,下拉选择<mark>前提条件</mark>中已创建的应用和环境,单击"新增组件"。

步骤3 参考表5-2配置组件信息。

表 5-2 nacos-consumer 组件基本信息

| 参数 | 说明 |
|------------------|--|
| 组件名称 | 新建组件的名称。本实践输入名称为"nacos-consumer"。 |
| 版本号 | 组件的版本号。 本实践版本号为1.0.0。 |
| 实例规格 | 选择实例规格,例如:0.5core、1GiB。 |
| 实例数量 | 输入实例数为1。 |
| 代码源 | 选择"源码仓库 > GitHub",配置授权信息。 • 授权信息: nacos-github。 • 用户名/组织: 用户授权使用的用户名/邮箱。 • 仓库名称: nacos-example。 • 分支: master。 |
| 语言/运行时 | 选择 "Java17" 。 |
| 自定义构建 | 选择"使用自定义命令"。 小黑框内输入: cd ./nacos-spring-cloud-example/nacos-spring-cloud-discovery-example/nacos- spring-cloud-consumer-example && mvn clean package |
| Dockerfile地 址 | 输入"./"。 |

- 步骤4 单击"配置组件"。
- 步骤5 在"组件配置"页面,单击"微服务引擎CSE"模块的"配置",进入"微服务引擎 CSE配置"页面。
- 步骤6 单击"注册配置中心",选择前提条件中您已创建的Nacos引擎。

注意: nacos-consumer组件和nacos-provider组件需绑定同一个Nacos引擎。

- 步骤7 单击"保存",完成对该组件微服务引擎CSE配置。
- **步骤8** 在"组件配置"页面,单击"环境变量"模块的"编辑",对nacos-consumer组件进行配置。
- 步骤9 单击"新增环境变量",输入变量名称及变量。
 - 变量名称: JAVA_TOOL_OPTIONS。
 - 变量/变量引用: --add-opens=java.base/java.lang=ALL-UNNAMED --addopens=java.base/java.lang.reflect=ALL-UNNAMED。

- 步骤10 单击"确定",完成对该组件环境变量配置。
- **步骤11** 在"组件配置"页面,单击"访问方式"模块的"编辑",进入"访问方式配置"页面。
- 步骤12 在"负载均衡配置"页签下,单击"添加负载均衡配置"并设置参数。
 - 协议:选择"TCP"。
 - 监听端口:输入"8080"。
 - 访问端口: 自定义访问端口。本实践设置为"14688"。
- 步骤13 单击"确定",完成对该组件访问方式配置。
- 步骤14 在"组件配置"页面,单击"配置并部署组件",等待组件部署完成。
- 步骤15 确认配置信息无误后,单击"确定",等待组件部署完成。

-----结束

调用微服务

- 步骤1 在左侧导航栏中选择"组件列表",进入"组件列表"页面。
- **步骤2** 选择nacos-consumer组件,在"访问地址"列单击□,获取自动生成的外网访问地址。

图 5-1 获取外网访问地址

| 访问方式 ⑦ | |
|----------------------------|----------|
| 从外部访问本组件: - http://100. | ☐ (公网地址) |
| | 编辑 |

步骤3 在浏览器输入访问地址:http://外网访问ip地址:访问端口/echo/{string},外网访问ip 地址、访问端口为<mark>步骤</mark>2中获取的,{string}为自定义字符串。例如:http:// 100.**.**.175:14688/echo/nacos-cae。

如果出现如下图所示欢迎页面,表示Nacos调用成功。

图 5-2 调用微服务引擎



Hello Nacos Discovery nacos-cae

----结束

6 Spring Cloud 应用无损上线

概述

在组件运维过程中,不可避免要进行升级、重启、扩容等操作,在这些操作中,无损 上线是常见的要求,本文介绍如何配置Spring Cloud无损上线。

前提条件

- 已**创建环境**。
- 已创建应用。
- 已创建并部署组件,确保您的组件为Spring Cloud。

操作步骤

步骤1 Spring Cloud 启用 spring-boot-starter-actuator。

1. 在组件对应的源码的pom.xml中添加spring-boot-starter-actuator依赖(建议使用 2.3.0及以上版本)。

<dependency>

- <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-starter-actuator</artifactId>
 <version>{您使用的spring boot版本}</version>
 </dependency>
- 2. 修改application.properties,添加如下配置:

表 6-1 application.properties 配置

| spring boot版本 | 配置 |
|---------------|---|
| 2.3.0/2.3.1 | management.health.probes.enabled=true |
| >= 2.3.2 | management.endpoint.health.probes.enabled=true management.health.livenessState.enabled=true management.health.readinessState.enabled=true |

- 3. 更新代码。
 - 若您的组件使用源码部署,请将修改的源码更新至源码仓库。

- 若您的组件使用软件包部署,请将新代码打包为软件包,并将新的软件包上 传至软件包仓库。
- 4. 使用新的源码或软件包,<mark>升级组件</mark>。

步骤2 配置组件健康检查。

- 1. 登录CAE控制台。选择"组件配置"。
- 2. 在"组件配置"页面上方的下拉框中选择待操作的组件。
- 3. 参考**配置健康检查**,配置就绪探针,具体参数如下:

表 6-2 就绪探针参数

| 配置内容 | 值 |
|--------|--|
| 检查方式 | HTTP请求检查。 |
| 端口 | 您的组件的实际监听端口。 |
| 路径 | spring boot 2.3及以上版本: /actuator/health/ readiness。 spring boot 2.3以下版本: /actuator/health。 |
| 协议 | 您的组件的实际协议。 |
| 检测周期 | 10 |
| 延迟时间 | 0 |
| 超时时间 | 1 |
| 成功阈值 | 1 |
| 最大失败次数 | 3 |

4. 升级组件,具体操作请参考<mark>升级组件</mark>。

步骤3 验证配置。

- 1. 选择"组件列表",若操作的组件状态为"运行中",则进入下一步,否则配置 失败。
- 选择"组件事件",在页面上方的下拉框中选择操作的组件。若存在"组件实例 健康"的事件,则配置成功,否则配置失败。

| 温馨提示:事件保存时间为1小时,1小时后自动调除救展 | | | | | | |
|----------------------------|--------|------|---------|---------------------------------|------------|------------|
| 事件名↓⊟ | 事件级别 🍞 | 業型 7 | 发生次数 ↓Ξ | 事件信息 | 首次发生时间↓Ξ | 最近发生时间 1= |
| 组件实例健康 | ◎ 正常 | 实例事件 | 1 | container docker://807829acb1d0 | 2023/10/23 | 2023/10/23 |

----结束



7.1 概述

健康检查用于检测您的应用实例是否正常工作,是用来保障业务正常运行的一种机制。CAE提供三种健康检查机制:存活探针、就绪探针和启动探针。

 存活探针用于检测应用程序是否存活。如果检测实例异常,k8s将会删除当前运行 的实例并重新检测,直到检测返回实例正常。

使用存活探针时需注意:

单独使用存活探针时,如出现网络波动或程序启动过慢的情况,会导致实例持续 被重启,且实例一直处于未就绪的状态。有如下解决方案:

- a. 与启动探针配合使用,具体操作请参见启动探针与存活探针配合工作。
- b. 使用时将"最大失败次数"调大,增加容错率,并增大"延迟时间",保证 程序在启动后再接受存活探针的检测。

须知

- 检查成功:对于在健康检查中设置的请求返回状态码200。
- 检查失败:对于在健康检查中设置的请求返回状态码非200,且连续失败次数 达到设置的"最大失败次数"。
- 就绪探针用于检测应用是否完成启动,并准备好开始接受请求。如果检查到实例
 已经健康,则进行流量的切换。

例如:以组件的实例从2个伸缩到3个为例,说明设置就绪探针前后对比。

a. 未设置就绪探针时,实例已创建完成,但由于程序原因,并未准备好开始接 受流量,此时如<mark>图7-1</mark>所示,部分流量仍然会进入实例3中。




b. 设置就绪探针后,就绪探针会进行检测,发现实例3并未启动完成或就绪,则 使实例3不接收流量,保证所有流量都会流入健康的实例1和实例2中。





c. 当就绪探针检查到实例3为健康时,则放通流量,停止本次检测。





 启动探针在运行时会禁用其他探测(就绪探针以及存活探针),如果启动探针检 查失败,则实例会被重启。
 启动探针建议搭配存活探针一起使用。

7.2 启动探针与存活探针配合工作

前提条件

- 已创建环境。
- 已创建应用。
- 已创建并部署组件,此示例中所用组件为demo-frontend。

操作步骤

- 步骤1 登录CAE控制台。选择"组件配置"。
- 步骤2 在"组件配置"页面上方的下拉框中选择待操作的组件。

图 7-4 选择待操作的组件

| ● 组件: demo-frontend-1698655497 / v1.0.0 | | 生效配置 6 ③ C |
|---|---|-------------------|
| 搜索 | Q | |
| emo-backend-1698655497 / v1.0.0 | | |
| 😔 demo-frontend-1698655497 / v1.0.0 | | |
| sada / v1.0.0 | | 微服务引擎CSE ⑦ |
| test0915 / v1.0.0 | | 您还未绑定CSE,如需使用请绑定。 |
| | | |
| | | |
| 配置 | | 配置 |

步骤3如**87-5**和**87-6**所示分别配置启动探针与存活探针,并生效配置,具体操作步骤请参考配置健康检查。

| <u>冬</u> | 7-5 配置存活探针 | | | | | | |
|----------|------------|--------|------------|---------|------|---|---|
| | 健康检查 | | | | | | × |
| | 存活探针 | 存活探针 🦲 | C | | | | |
| | 就绪探针 | 检查方式 | HTTP请求检查 | TCP端口检查 | 执行命令 | ? | |
| | 启动探针 | 端口 | 8080 | |] | | |
| | | 路径 | 例: /path/a | |] | | |
| | | 协议 | HTTP | HTTPS | | | |
| | | 检测周期 | 10 | | 秒 | | |
| | | 延迟时间 | 0 | | 秒 | | |
| | | 超时时间 | 1 | | 秒 | | |
| | | 成功阈值 | 1 | | | | |
| | | 最大失败次数 | 3 | | 次 | | |
| | | 请求头 | ⊕ 添加请求头 | | | | |

图 7-6 配置启动探针



启动探针会在实例创建后10秒钟进行检查,每5秒检测一次,且连续失败5次后进行容 器重启的操作。在启动探针检测到实例健康后,存活探针将进行启动检测,可以避免 程序启动过慢,导致实例一直重启。

使用启动探针时需注意:请保证程序启动时间在"延迟时间"+"检测周期"*"最大 失败次数"秒内能启动,否则启动探针会一直重启实例。如果不确定程序启动时间, 建议调大"最大失败次数"以及"延迟时间"。如图7-6所示,程序需要在10+5*5 = 35秒内启动。

步骤4 单击"组件事件",在"组件事件"页面中可以看到启动探针探测一次失败后,程序 成功启动,切换到了存活探针,并检测实例健康。

图 7-7 查看事件信息

| 9 組件: demo-frontend-1698 | 655497 / v1.0.0 🔻 | | | | 词输入 | 、事件名进行搜索 Q C |
|--------------------------|-------------------|------|--------|--|--------------------------|--------------------------|
| 温馨提示: 事件保存时间为1小时 | , 1小时后自动清除数据 | | | Startup probe failed: Get | | |
| 事件名↓⊟ | 事件级别 🍞 | 类型 7 | 发生次数 🕽 | "http://10 3080/": context deadline exceeded (Client.Timeout exceeded while | 自次发生时间 ↓Ξ | 最近发生时间 ↓ |
| 组件实例健康 | ☺ 正常 | 实例事件 | | awaiting headers) | 023/10/30 17:24:27 GMT+ | 2023/10/30 17:24:27 GMT+ |
| 组件实例不健康 | 9 异常 | 实例事件 | | 1 Startup probe failed: Get "http://10.0.1.6 | 2023/10/30 17:24:23 GMT+ | 2023/10/30 17:24:23 GMT+ |
| 组件启动 | ◎ 正常 | 实例事件 | | 1 Started container demo-frontend-16986 | 2023/10/30 17:24:08 GMT+ | 2023/10/30 17:24:08 GMT+ |

图 7-8 启动探针探测情况

| ● 组件: demo-frontend-1698 | 855497 / v1.0.0 🔹 | | | | 请输入 | 事件名进行搜索 | QC |
|--------------------------|------------------------|------|-------|--|--------------------------|-----------------------|------|
| 温馨提示:事件保存时间为1小时 | , 1小时后自动清除数据 | | | container docker://2bf563c9a2eec7aa3b90ce47a4dedf 4c13c5dc2ftb/5e5ee5c74c64dcf337d03 in health status | | | |
| 事件名↓Ξ | 事件级别 🍞 | 类型 🍞 | 发生次数、 | 12 | 自次发生时间 ↓Ξ | 最近发生时间 15 | |
| 组件实例健康 | ◎ 正常 | 实例事件 | | 1 container docker://2bf563c9a2eec7aa3b | 2023/10/30 17:24:27 GMT+ | 2023/10/30 17:24:27 0 | 9MT+ |
| 组件实例不健康 | 0 异常 | 实例事件 | | 1 Startup probe failed: Get "http://10.0.1.6 | 2023/10/30 17:24:23 GMT+ | 2023/10/30 17:24:23 0 | €MT+ |
| 组件启动 | 正常 | 实例事件 | | 1 Started container demo-frontend-16986 | 2023/10/30 17:24:08 GMT+ | 2023/10/30 17:24:08 0 | GMT+ |

----结束

7.3 使用就绪探针保证升级时流量正常

前提条件

- 已创建环境。
- 已**创建应用**。
- 已创建并部署组件,此示例中所用组件为demo-frontend。
- 已配置访问方式并生效配置。

操作步骤

步骤1 登录CAE控制台。选择"组件配置"。

步骤2 在"组件配置"页面上方的下拉框中选择待操作的组件。

图 7-9 选择待操作的组件



步骤3 如图7-10所示配置就绪探针,并生效配置,具体操作步骤请参考配置健康检查。

| 存活探针 | 就绪探针 | \bigcirc | | | |
|------|--------|------------|---------|------|---|
| 就绪探针 | 检查方式 | HTTP请求检查 | TCP端口检查 | 执行命令 | ? |
| 启动探针 | 端口 | 8080 | | | |
| | 路径 | 例: /path/a | | | |
| | 协议 | нттр | HTTPS | | |
| | 检测周期 | 10 | | 秒 | |
| | 延迟时间 | 0 | | 秒 | |
| | 超时时间 | 1 | | 秒 | |
| | 成功阈值 | 1 | | | |
| | 最大失败次数 | 1 | | 次 | |
| | 请求头 | ⊕ 添加请求头 | | | |
| | | | | | |

图 7-10 配置就绪探针

步骤4 对组件进行升级操作,具体操作请参考升级组件。

步骤5 升级时,在左侧导航栏中选择"组件列表",进入"组件列表"页面,单击对应组件 "访问地址"列的ip地址,查看应用页面。如<mark>图</mark>7-11所示服务并无中断。

图 7-11 访问应用页面



步骤6 在"组件事件"页面,查看组件事件,可以看到实例存在不健康的请求,此时检测到 新实例还未准备好进行流量切换,则继续使用旧实例提供服务。

图 7-12 查看组件事件

| 55. (2117)(5 | BINE: demo-frontend-1991639208 / v1.8.1 • | | | | | | | The Alteration of C |
|-----------------|---|------------------------|------------------|------|----|---|-------------------------------|-------------------------------|
| 20938 | 這種情示。專件很好时间为1小时,1小时也是改變的設備 | | | | | | | |
| 经计算管 | 単作名 (三 | augu A | 美型 刀 | 凝生次数 | 48 | Pridde | E0312100 (E | BREACTORN IS |
| 0180 | 協作の対理事 | 0 IFM | 实代事件 | | 1 | Container docker 7459-77-06-131-8593-7-06-1020 215171663542 10.0.13.18593-7-06-1020 215171663542 | 2023/08/12 11:31:32 GMT+08:00 | 2023/08/12 11:31:32 GMT+08:00 |
| inia-7.2 | 04400.000 | 正常 | (2)+草(+ | | 1 | (combined from refused and-1991638 | 2023/08/12 11:31:32 GMT+08:00 | 2023/08/12 11:31:32 GMT+08:00 |
| sitt22 | 1894年6月不健康 | 0 35 | (2) | | 1 | Readmoss probe failed: Out "Http:/10.0.0.131.80001" dial top 10.0.0.131.8080.co | 2023/08/12 11:31:24 OMT+08:00 | 2023/08/12 11:31:24 GMT+08:00 |
| | 经卷成功 | 0 IIN | 采約事 件 | | 2 | Successfully mounted volumes for pod "demo-frontend-1691636280-605647 cdb9 | 2023/08/12 11:31:22 GMT+08:00 | 2023/08/12 11:31:24 GMT+08:00 |
| | (1)(年前23) | • EM | 元 內華州 | | 1 | Started container demo-frontend-1091939288 | 2023/08/12 11:31:23 GMT+08:00 | 2023/00/12 11:31:23 GMT+00:00 |
| | (1)年最新計量成2) | 0 EM | 2018年 | | 1 | Created container domo-frontend 1691638288 | 2023/08/12 11:31:23 GMT+08:00 | 2823/88/12 11:31:23 GWT+08:00 |
| | 10008400 | 正常 | 实的事件 | | 1 | Container image "twicon-north-7 myhuaweicloud comipaes_coe_lwv410940_011c | 2023/08/12 11:31:23 GMT+08:00 | 2023/08/12 11:31:23 GMT+08:00 |
| | (0450)BR | 正常 | \$19 8 14 | | | Successfully assigned 19842e151e75498878a911e62544c3e-lesi-wurdento-frent. | 2023/08/12 11:31:22 OMT+08:00 | 2023/08/12 11:31:22 GMT+08:00 |
| | 01420300 | 0 IN | 814 8 14 | | 1 | Scaled up replica set demo-frontend-1991639288-865447c499 to 1 | 2023/08/12 11:31:22 OMT+08:00 | 2823/08/12 11:31:22 GMT+08:00 |
| | 0123/049 | 0 EM | (2)10年10 | | 4 | Scaled down replica set demo-frontend-1691639280-5474578799 to 0 | 2023/00/12 11:31:22 GMT+08:00 | 2023/00/12 11:31:22 GMT+08:00 |
| | 10 ¥ 888:17 < 1 2 > | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |

----结束

8 生命周期管理

8.1 概述

生命周期管理是用于在特定阶段执行调用的方法。CAE提供两种生命周期管理:启动 后处理、停止前处理。

 启动后处理:组件实例启动后立即触发启动后事件,但是不确保对应的handler是 否能在容器的EntryPoint之前执行。

🛄 说明

只有启动后处理函数执行完毕,组件实例的状态才会变成Running。因此当命令设置为死循环时,CAE组件状态将无法正常变成运行中。

• 停止前处理:组件实例停止前,会发送停止前事件。注意:

🛄 说明

组件实例结束前会立即发送停止前事件,除非实例宽限期限超时,组件实例会一直阻塞等 待停止前函数执行完毕。

8.2 利用启动后处理写文件

前提条件

- 已创建环境。
- 已**创建应用**。
- 已创建并部署组件,此示例中所用组件为nginx组件。

操作步骤

步骤1 登录CAE控制台。选择"组件配置"。

步骤2 在"组件配置"页面上方的下拉框中选择待操作的组件。

| 图 | 8-1 | 选择组件 |
|---|-----|----------|
| 5 | 0-1 | 201千1111 |

| 云应用引擎 | び 环境: ● env-test ▼ ⊕ 廠 应用: | fecycle 💌 🕀 |
|-------|--|-------------------|
| 概览 | | |
| 组件列表 | ● 组件: nginx / v1.0.0 | |
| 实例列表 | 山间性 | |
| 组件配置 | דונייויד | |
| 组件事件 | 云数据库RDS ⑦ | 微服务引擎CSE ⑦ |
| 组件监控 | 您还未绑定云数据库RDS,如需使用RDS请绑定。 | 您还未绑定CSE,如需使用请绑定。 |
| 组件日志 | | |
| 系统设置 | • | |
| | 置頃 | 配置 |

步骤3 如图8-2所示配置启动后处理,具体操作请参考配置生命周期。

分别输入以下命令: /bin/bash -c echo 'Hello, postStart' > /lifecycle.txt

图 8-2 配置生命周期

生命周期管理

| 启动后处理 | ! (PostStart设置) | 停止前处理 | (PreStopi | 受置) |
|-------|------------------------------|--------------|-----------|-----|
| | | | | |
| | 2099年1月1112日1月20月11日118日20日 | | | |
| 处理方式 | 命令行脚本 | | | |
| 执行命令 | /bin/bash | | | Θ |
| | -c | | | Θ |
| | echo 'Hello, postStart' > /l | ifecycle.txt | | Θ |
| | 新增执行命令 | | | |

步骤4单击"确定",保存配置。

步骤5 单击页面上方"生效配置"。在右侧弹框中确认配置信息,并单击"确定",使配置 生效。

步骤6 使用CloudShell登录容器,查看文件内容是否生效。

- 1. 在左侧导航栏中选择"实例列表"。
- 2. 在"实例列表"页面上方的下拉框中选择环境、应用及待操作的组件。
- 选择运行中的实例,在"操作"列单击"远程登录"。进入容器内部,查看文件 内容是否生效。
- 4. 查看文件内容,确认是否已写入成功。 tail -f lifecycle.txt

图 8-3 登录容器查看文件内容

| CloudShell 🧔 🞯 🔱 🖹 | |
|----------------------------------|--------------------------------------|
| >_ user(| |
| user@lrbd7g7hd79mzbw-machine:~\$ | kubectl -n 79c408 |
| root@nginx-stable-alpine-1 | n:/# ls |
| bin boot dev etc home lib | lib64 lifecycle.txt media mnt opt |
| root@nginx-stable-alpine- | <pre>:/# tail -f lifecycle.txt</pre> |
| Hello, postStart | |
| | |
| | |
| | |
| | |

----结束

8.3 利用停止前处理优雅关闭 Nginx

如果容器碰到问题被系统关闭,停止前处理可以帮助您的主程序在关闭前执行必要的清理任务。

前提条件

- 已创建环境。
- 已创建应用。
- 已创建并部署组件,此示例中所用组件为nginx组件。

操作步骤

- 步骤1 登录CAE控制台。选择"组件配置"。
- 步骤2 在"组件配置"页面上方的下拉框中选择待操作的组件。

| 图 8-4 | 选择组件 |
|-------|------|
|-------|------|

| 云应用引擎 | ₩ 环境: • env-test • (+) (km | rcle • + |
|-------|---|-------------------|
| | | |
| 概览 | | |
| | ● 组件: nginx / v1.0.0 | |
| 组件列表 | | |
| 实例列表 | 中间件 | |
| 组件配置 | דתפויד | |
| 组件事件 | 云数据库RDS ⑦ | 微服务引擎CSE ⑦ |
| 组件监控 | 您还未绑定云数据库RDS.如需使用RDS请绑定。 | |
| | | 您还未绑定CSE,如需使用请绑定。 |
| 组件日志 | | 您还未绑定CSE,如屬使用頭绑定。 |
| 组件日志 | 4 | 您还未绑定CSE,如鬻使用商绑定。 |

步骤3 如图8-5所示配置停止前处理,具体操作请参考配置生命周期。

| 分别输入以下命令: /bin/bash -c nginx -s quit;while killall -0 nginx;do sleep 1;done | |
|--|-------------------|
| 图 8-5 配置生命周期 生命周期管理 | |
| 启动后处理 (PostStart设置) | 停止前处理 (PreStop设置) |

| 0 | 停止前处理将在应用停止 | (Terminated) | 前触发, | 应用因执行结束正常退出 |
|---|-------------|--------------|------|-------------|
| | | | | |

处理方式 命令行脚本

| 执行命令 | /bin/bash | Θ |
|------|--|---|
| | -C | Θ |
| | nginx -s quit;while killall -0 nginx;do sleep 1;done | Θ |

分 新增执行命令

步骤4 单击"确定",保存配置。

1

步骤5 单击页面上方"生效配置"。在右侧弹框中确认配置信息,并单击"确定",使配置 生效。

----结束

9 发送事件告警到企业微信

概述

CAE支持在实例调度成功/失败,健康检查成功/失败,拉取镜像成功/失败,卷挂载成 功/失败,容器启动成功/失败时发送通知。通过设置事件通知规则,可以帮助您及时了 解组件运行时的状态,快速定位问题。配置事件通知规则依赖AOM服务和SMN服务, 组件实例事件上报AOM服务,您可以通过"AOM > 告警管理 > 告警列表 > 事件"查 看上报的事件,SMN服务是实际的告警消息的发送方。

CAE默认上报组件实例事件到AOM,用户在配置事件通知规则之后,由CAE在SMN服务创建主题、添加订阅者,在AOM服务创建告警规则、告警行动规则,完成整个事件上报告警流程的配置。



图 9-1 事件上报告警流程

前提条件

SMN服务支持添加企业微信群消息、钉钉群消息、飞书群消息的订阅,目前,这些功能处于公测期间,您需先申请开通SMN服务的公测资格。

- 已开通企业微信,并创建了群机器人。
- 已在华为云提交工单申请SMN服务的公测资格,并已开通公测。

操作步骤

步骤1 登录CAE控制台,选择"系统设置"。

步骤2 单击"事件通知规则"模块中的"编辑",进入"事件通知规则配置"页面。

步骤3单击"创建事件通知规则",参考表9-1配置基本信息。

表 9-1 配置基本信息说明

| 配置项 | 配置项说明 |
|--------------|---|
| 事件通知规则名 称 | 输入事件通知规则名称。例如:container-Initiate。 由英文字母、数字、中划线和下划线组成,并以英文字母开头和 结尾,长度为1-64个字符。 |
| 触发事件 | 在下拉框中选择事件通知的触发事件。例如:容器启动成功。 |
| 生效组件范围 | 选择环境内所有组件。 |
| 触发策略 | 触发方式选择"立即触发"。 |

步骤4 选择通知方式为"企业微信机器人"。

填写企业微信机器人终端地址:请输入以https://qyapi.weixin.qq.com/cgi-bin/webhook/send开头的webhook地址。

🛄 说明

获取企业微信订阅终端参考企业微信机器人如何获取订阅终端。

图 9-2 配置事件通知

| 事件通知 | | |
|------|--|---|
| 通知方式 | 企业微信机器人 邮件 短信 | |
| | A 机器人协议目前处于公测阶段, 需用户提工单申请开通。工单链接 | × |
| 终端 | https://qyapi.weixin.qq.com/cgi-bin/webhook/send?key=c | |
| 正文 | 通知类型: \${event_type}; 事件投题: \${event_severity}; 事件名称: \$event_metadata.event_name; 发生时间: \${starts_at}; 事件源: \$event_metadata.resource_provider; 资源频型: \$event_metadata.resource_type; 资源标识: \${resource_new}; 可能原题: \${resource_new}; 可能原题: \${alarm_probableCause_zh}; 附加信息: \${message}; 修复建议: \${alarm_fix_suggestion_zh}; | |

步骤5 单击"确定",完成事件通知规则创建。

步骤6 在组件列表中选择待操作组件,单击操作列"更多 > 重启"。

等待组件重启成功后,您可以登录AOM控制台查看事件列表或在企业微信群接收到告 警消息。

- 登录AOM控制台查看CAE上报的事件列表,
 - a. 选择"告警管理 > 告警列表 > 事件"。
 - b. 事件级别勾选"全选"。
 - c. 选择筛选条件为"事件源:CAE",单击 Q 按钮。

图 9-3 事件列表

| 〇 点用运输器 | 118 <mark>812</mark> | All 2022/06 | | ● 目动化运输 采集管理 | | | C STREET S ACAN STREET |
|--------------------------|----------------------|--|---|-----------------------------|---|--|-------------------------------------|
| 229 0 0 | | 212 4/1 0 | | | | | 0 |
| 線入中心 | | ##\$25:55;22(27)570 | ■#4妻: CAE ◎ 活出的运会/4 | | | | × Q 0 1510994 . 0 • |
| 位非直 | | | | | | | • • |
| 会留管理 L martine | | 25 C 3CB 0 C 3EW 9 | 0 mm 0 0 mm 52 | | | | |
| 2010 | • | 20 15 | | | | | |
| 体包线权 | | 5 | | | | | |
| 11121210 | | 17/17 17/47 18/17 18/47 | 1917 1947 2017 2047 2117 2147 22 | 17 2247 2317 2347 0017 0047 | 01117 0147 0217 0247 0317 0347 0417 0447 051 | 7 8567 8617 8667 87117 8767 8617 8647 8917 9967 1017 1667 1117 1167 1217 1247 111 | 7 1347 1447 1447 1547 1547 1647 164 |
| 000720601 | | PRSB | 941(53) | 中仁市 完建 实型 | 9250 | | MXEN |
| 应用用制 如器用用 | | 实例编数成功 | 0 遗示 | CAE component | CAE env north4.app.demo-backend-1694722178 | CV2: env north4 app 6eme backeds 1994722178: deene backerds 664722178- teosearrow aun CV2: II Backs Johl Sch. 1915/10: 595100 | 2023/09/15 10:17:39 GMT+08:00 |
| 14100月 新社会同 <u>法刊</u> | | 機會投影成功 | ○ 證示 | CAE component | CAE.env-north4.app.demo-backend-1694722178 | Container image "switch-north-4 myfruaweicloud com/mylearning-servicecombicae-backend v1.0.1" alle | 2023/09/15 10:17:39 GMT+08:00 |
| Prometheus <u>232</u> | | 容器包达成功 | ○ 提示 | CAE component | CAE.env-north4.app.demo-backend-1694722178 | Started container demo-backend-1694722178 | 2023/09/15 10:17:39 GMT+08:00 |
| 日本分析 | • | 塑挂载成功 | ◎ 推示 | CAE component | CAE.env-north4.app.demo-backend-1694722170 | Successfully mounted volumes for pod "demo-backend-1694722178-6656469768-dzst_7ca2ab2421e9 | 2023/09/15 10:17:39 GMT+08:00 |
| 和 描订词 | | 容量的功能功 | 0 现示 | CAE component | CAE, env-north 4 app. demo-backend-1694722176 | Stanted container demo-backend-1694722178 | 2023/09/15 10:07:39 GMT+08:00 |
| | | 機動控制成功 | ◎ 提示 | CAE component | CAE.env-north4.app.demo-backend-1694722176 | eq:successfully pulled image 'www.cn-north-4.myhuaweldoud.com/myleaming-servicecomb/cae-backend.v | 2023/09/15 10:07:38 GMT+08:00 |
| | | 着挂载成功 | ◎ 提示 | CAE component | CAE.env-north4.app.demo-backend-1694722178 | Successfully mounted volumes for pod "demo-beckend-1694722176-6655469768-kt/v8c_Toa2a6242149 | 2023/09/15 10:07:26 GMT+08:00 |
| | | 实例隔宽成功 | ◎ 提示 | CAE component | CAE. env-north4 app. demo-backend-1694722178 | Successfully assigned 7ca2at2421e94503a410ab47685a25d2-env-notth4/demo-backend-1694722178 | 2023/09/15 10:07:25 GMT+08:00 |
| | | 安何藏宽成功 | ○ 總示 | CAE component | CAE.env-north4.app.demo-backend-1694722178 | Successfully assigned 7ca2a82.421e545503a410ab47685a25d2-env-not7x438emo-backend-1654722178 | 2023/09/15 10:05:32 GMT+08:00 |
| | | 安何羅変失敗 | 0 EF | CAE component | CAE.env-north4.app.demo-backend-1694722178 | 0.4 nodes are available: 2 insufficient memory, 2 node(s) had taint (cae-system: true), that the pod didn't | 2023/09/15 10:05:32 GMT+08:00 |
| | | 10 · · · · · · · · · · · · · · · · · · · | 2 3 7 > | | | | |

• 在企业微信群接收到告警信息。

| 图 9-4 | 告警信息 |
|-------|------|
|-------|------|

| ÷ | (11) | <u>.</u> . | : |
|---|---|---------------|---|
| | 更多信息请登录CAE。 | | |
| | 以下为新消息 | | |
| | 16:23 | | |
| á | | | |
| | 告警名称: 容器启动成功; 告警ID: 4280570913916852605; 告警时间: 2023-09-15 16:23:30 GMT+08:00; 告警行动规则: cae-test-weixin-5ef7; 告警级别: 提示; 告警内容: Started containe test-demo-1; 资源标识: CAE.env-test.app.test-de o-1.test-demo-1-8646f74 cd-fx9b6(CAE.环境名称.应 | er m l8 | |
| | 名称.组件名称.头例名称); ; | | |
| | 修复建议:NA; 本消息是由 E华之 云应用运维管理AOM中设置 告警规则容器启动成功触发; 更多信息请登录CAE。 | 为的 | |

----结束

10 对接软件开发生产线 CodeArts 流水线 自动升级到 CAE

10.1 概述

CAE目前提供了Codearts商业插件,可对接Codearts流水线自动升级组件到CAE,此章 节将根据不同场景为您介绍"CAE升级插件"的使用方法。

目前支持从Codearts构建为镜像,再部署到CAE,也可以通过Codearts构建为软件包 上传至Codearts的软件发布库或OBS中,再部署至CAE。

前提条件

- 已开通软件开发生产线CodeArts服务,已创建项目并托管代码。
- 已创建组件。

10.2 流水线构建软件包上传到 Codearts 软件发布库后升级 CAE 组件

创建编译构建任务

- 步骤1 进入CodeArts控制台,单击右上角"立即使用"。
- 步骤2 在菜单栏选择"服务 > 编译构建"。

| ···· 华为云 控制台 | 首页 黃 工作台 學 看板 | 3 服务▼ 📕 | 华为开源镜像站 | | | 4 0 |
|--------------|------------------|---------|---------|-----------|------------------|--------|
| ✿ 我的关注 | | ch | | C. New | 4 | |
| 所有项目 | Hello, | 代码托管 | 代码检查 | IDE | 流水线 | 编译构建 |
| 音 已归档 | 您还没有项目,从创建一~ | î | | | | |
| | | - | ۷ | 10 A | | - ≼ |
| | 项目模板 | 部署 | 扩展插件 | 制品仓库 | 测试计划 | 移动应用测试 |
| | RR A | 3 | | \sim | 1 | 3 |
| | | 文档 | Wiki | DevStar ≫ | APIOps <i></i> > | 漏洞管理ス |
| | AR | | | | | |
| | IPD-系统设备类 51/5 0 | 开源治理> | | | | |

步骤3 在编译构建页面单击"新建任务",在基本信息中填入"任务名称",并选择"归属项目"、"源码源"、"源码仓库"和"分支",单击"下一步"。

图 10-2 新建任务

| HUAWT | 华为云 | 控制台 | 0 | • • | 💧 首页 | 🚔 工作台 | 🛃 管理看板 | ◎服务▼ |
|-------|------------------------|---------------|-------|-----|------|-------|--------|------|
| | 编译构建 成功率 54% | 6 任务个数 | 改 249 | | | | | |
| +新 | 建任务(| 〉 搜索任务 | 名称 | | | | | |

图 10-3 配置基本信息

| ← ## | | | ig ² á | 新建构建任务 | | | |
|--------------|--------------------------|---------------------|-----------------------|-------------------|------------|--|--|
| Sature State | | | | | | | |
| ③ 构建螺板 | 基本信息 | | | | | | |
| | •任务名称 | | | | | | |
| | cae-backend-softwa | ire-release-library | | | | | |
| | • 归期项目 | | | | | | |
| | cae-teset | | | | Ψ. | | |
| | 連結連 | | | | | | |
| | CA Repo | GitHub | ● 通用Git | GitCode | (5) | | |
| | GitLab | Gernt | 《》 其他项目Repo | 。 来自流水线 | | | |
| | 源码仓库 | | | | | | |
| | cae-backend | | | | - | | |
| | • 分支 | | | | | | |
| | master | | | | | | |
| | 任务描述 | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | 0/512 | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | Ę | 0H T-# | | | |

步骤4 选择构建使用的模板(此示例使用的为java代码,故此步选择Maven模板)后,单击"下一步"。

图 10-4 选择构建模板

| | | | | 💕 新建构建任务 | |
|--------|----------|----|-------|--|---------------------|
| ✓ 基本信息 | | | | | |
| 約建模板 | 构建模板 | | 推荐模板 | | 1 |
| | Q 搜索模板名称 | | Maven | Maven 使用Apache Maven构建Java项目 | 0 |
| | 心 推荐模板 | 1 | _ | | |
| | 昭 公共模板 | 37 | 公共模板 | | |
| | A。自定义模板 | 0 | | 空白梅建模板 使用系统构建步骤自由定义构建任务。 | |
| | | | * | Service Stage-Maven-積像构建 使用Apache Maven构建Java项目,通过Dock | erfile制作编编并推送到SWR合库 |
| | | | | Service Stage-Maven-jar包构建 使用Apache Maven构建Java项目,打成jar包。 | 上传到没有库 |

步骤5 在"参数设置"页面,"自定义参数"中单击"新建参数",并开启运行时设置。

- 名称: 输入release_version。
- 值:取值使用时间戳变量: \${TIMESTAMP}。

图 10-5 添加自定义参数

| ← cae-backend-software-release-library 基本信息 | 构建步骤 参数设置 执 | 行计划 橡放历史 权限管理 通知 | | | |
|---|-------------|------------------|--------|---------|--------------|
| 自定文参数 系统预定义参数 Q 训输入关键学 | | | | | |
| 名称 | 奥型 | 默认值 | 私密参数 💿 | 运行时设置 💮 | 仲政描述 |
| codeBranch | 字符串 | master | | | 代码分支,系统预定义参数 |
| release_version | 字符串 | \${TIMESTAMP} | | | |
| + 新建参数 | | | | | |

步骤6 进入"构建步骤"页面后,选择"上传软件包到软件发布库","发布版本号"选择 使用变量"\${release_version}"作为软件包存储的目录。

配置固定的包名,存储在软件发布库中,单击"新建并执行"。

图 10-6 上传软件包

| \leftarrow cae-backend-software-release-l | library 基本信息 构建步骤 参数设置 执行计划 修改历史 | ・ 収限管理 通知 |
|---|---|---|
| 國形化 | 代码化 化成环场化器 新期间间在全面间下环期 | ■ 上传软件包到软件发布库 上传软件包到软件发布库 食着进行脑南 |
| یان Mave | 代明 F 転配 置 和面代明 F 転の方 の MavenR4目 | ・ 伊藤臣示志称 上代松林島敏次作芝有岸 * 均陸島政会 ● **/target* 7ar 五石鉱長今 ● |
| | ②田均4454 Marenet(風上水2515. 上付25月13日(日本)3555 | ିହ(release_version) ଅନ୍ଧ 🔘 i cae-backend |

步骤7 确认运行时参数后,单击"确定"。

图 10-7 配置运行时参数

| 执行 | | × |
|------------------------|--------------|----|
| 运行时参数 ^{名称} | 值 | |
| release_version | \${TIMESTAME | 2} |
| | 确定取消 | |

步骤8 查看构建日志,构建成功并上传软件包到软件发布库中。

| 图 10-8 | 「「「「「」」 | 构建日志 |
|--|-----------------------|--|
| 首页 / cae-teset / 编译构建 / 构 | 建历史 / ci | ae backand-software-release-library |
| ← ♥ #20231028.13 代码源: <i>☆ cae-backend</i> | 8 ⁹ master | NURF##: FINEX 标题的序句时形: 圖 20231926 20 07 35 GMT-46.00 ①56a 关联: 合 1 标题 Fill |
| 步骤日志 构建参数 | | |
| ✓ 代码检出 | 7s | 构建日志 |
| ✓ Maven构建 | 235 | 1628 [2023/10/28 20:08:23.131 GHT+03:00] as: COMUT_ID_SHORTER#281921a 1629 [2023/10/28 20:08:23.131 GHT+03:08] as: COMUT_ID=283921aa693602258e145aa537ccrf2#7810#718 |
| ♥ 上传软件包别软件发布库 | 75 | 12021/10/28 200623.111 0641-0861 part (00001102) 06001102/10012011021 12022/10/28 200623.112 0641-0861 part (00001102) 060011020100000000000000000000000000000 |
| | | 1656 (2023/20/28 20:08:23.33/ 001-08:00)[IMFO] [L作校订世的双叶方の生:external_nexus_artifact_upicader] : 方面的包括/H:target/cae-demo-1.0-SMAPSHDT.jar 1637 [2023/10/28 20:08:23.639 GHT+08:00] [IMFO] [上作软件包的软件发布度:external_nexus_artifact_upicader] : 文件 target/cae-demo-1.0-SMAPSHDT.jar 板边成功 1638 |
| | | 1007 [2023/10/28 20:08:25.780 GHT+08:00] [1NFO] [上传软件包则软件发布算:external_nexus_artifact_uploader] : 开始上传文件[target/cae-demo-1.8-5N4PSH0T.jar至仓盘路径]/cae-backend/2023302820 1640 |
| | | 1641 [2023/10/28 20:08:29 551 Off-08:00] [INFO] [上作软件型为第isstermal_nexus_artifact_uploader] : outputs {"cae-backend.jar":{"name":"cae-backend.jar":"20231022020733"; "url":"/cae-backend.jar":{"name":"cae-backend.jar":?" |
| | | (42) (2021) 2012 2016 (2014) 2014 (2014) (1004) [Leff R # B 数 f L = K = K = K = K = K = K = K = K = K = |
| | | 527 [20110023 308129.931 GHT-86 10] [1040] [1月前日最短期日本第年1047943], and pure strain and pure st |

步骤9 在"制品仓库 > 软件发布库"中查看上传的软件包,目录结构为:项目名称-包名-构建时间戳-软件包名称。

图 10-9 查看软件包

| 软件发布库 博输入制品名称搜索 | 第四 / Cae-teset 和回 / Cae-teset 和回来型 generic 創業者 会議時1月 2023/09/10 18.41.14 GMT-00.00 合用無法 | |
|---------------------------|---|--------------------|
| TE 仓库视图 	 Ξ 版本视图 | cae-teset / cae-backend / 20231028200733 / cae-backend.jar | |
| cae-teset | 截流 构建元数据构建包码档制品安全 | |
| | 分半系称 cae-teset 相対指照 (cae-backend/20231028200733/cae-backend jar □ <u>软件包下载地址</u>) 下数地址 https://devrapo.c 文元版版本 20231028200733 / | :0231028200733/cae |

-----结束

创建流水线升级 CAE 组件

创建Codearts流水线:

步骤1 返回Codearts控制台首页。

步骤2 在菜单栏选择"服务 > 流水线",在"流水线"页面单击"新建流水线"。

文档版本 01 (2025-01-22)

图 10-10 选择流水线

| ₩ 华为云 | 控制台 | ♀ ◆ ● 首页 片 工作台 型 管理看板 | ◎ 服务 • | | | | |
|-------------|-----|---------------------------------|---------|------------|-------------|-----------|-----------|
| cae-teset | | 首页 / cae-teset / 流水鏡 | | | | | |
| T /m | | 流水线 策略 | - 22 | \$ | | 4 | |
| LTF | , | 금 ▲ 全部分组 + 新建流水线 全部流水线 ▼ Q 请输入器 | 架构建模 | 代码托管 | 代码检查 | 流水线 | 编译构建 |
| 剁构建模 | | 名称 | | | | | |
| 七码 | > | #26 car | | | 4 | 4 | 3 |
| 寺续交付 | ~ | Cae-demo P maste | 测试计划 | 部署 | 制品仓库 | Wiki | 文档 |
| 流水线 | | | | | | | |
| 发布管理 | | | ۷ | - | New | 512 | |
| 编译构建 | | | 扩展插件 | IDE Online | IDE | DevStar A | APIOps >> |
| 部署 | | | | | | | |
| 运维 | | | 0 | 0 | Flow | | |
| 制品仓库 | > | | 漏洞管理服务习 | 开源治理服务》 | Astro 工作流 > | | |
| BR≠ | | | | | | | |

图 10-11 新建流水线

| 🔌 | 🌿 华为云 | 控制台 | ♥ ▼ 意首页 異 工作台 | 管理看板 🔹 服务 🕶 | |
|-------------|---------|--------------|--------------------------|-------------------------------|-------------|
| C 😋 | e-teset | • | 首页 / cae-teset / 流水鏡 | | |
| р т | -v= | | 流水线 策略 | | |
| ··· ·· | _ (F | [^] | ∃ 🖹 全部分组 + 新建流水线 全部流水线 👻 | Q 请输入流水线名称 | |
| 9 9R | 时间观察作用 | | 各称 | 最近一次执行 | 工作流 |
| 新 代 | (49) | > | | #26 cae-backend 更新 Dockerfile | |
| 🍮 持 | 操交付 | ~ | Cae-demo | ₽ master | 0- 0 |
| 5 11 | 流水线 | | | | |
| # 2 | 发布管理 | | | | |
| و ی | 编译构建 | | | | |

步骤3 在"新建流水线"页面,输入"名称"并选择"流水线源"、"代码仓"和"默认分支",并单击"下一步"。

| 图 10-12 配置流水线基本 | 信息 |
|-----------------|----|
|-----------------|----|

| @ 基本信息 | | |
|------------|--|-------|
| (T) 188287 | 基本信息 | |
| U 1444 | *所属项目 | |
| | cae-teset 👻 | |
| | *石称 | |
| | software_release_library_pipeline | |
| | *語水起源 | |
| | ♀ ♀ ♀ ↓ Repo GitHub ∅∃⊆Git Gerrit GitLab | |
| | 🔶 G 🚸 🔲 | |
| | *代四仓 期355 | |
| | cae-backend 👻 | |
| | *默以分支 @ | |
| | master 👻 | |
| | Repo https接权 ③ 新建操权 | |
| | 遺造率 | |
| | 别名 💿 | |
| | 别名仪支持输入大小写英文字母、数字、二,最多128个字符 | |
| | 描述 | |
| | apartijic/stelles | |
| | | 0/200 |
| | | |
| | | |
| | | |

步骤4模板选择"无>空模板",单击"确定"。

| 基本信息 | 模板选择 | | 自定义模板 | 18新建模板 這收起 |
|------|---------|----|------------------------------|------------|
| | Q 搜索模板 | | template-20211026153921 | |
| | 88 全部 | 20 | 开始构建结束 | |
| | 🛃 Java | 8 | | |
| | Node.js | 5 | ServiceStage-Maven-虚机部署-copy | |
| | Python | 2 | 开始 代码检查 构建 部署 结束 | |
| | ∞ Go | 1 | 系统模板 | |
| | • PHP | 1 | 空模板 | 0 |
| | ↔ 其他 | 1 | | |
| | a Ŧ | 2 | 开始 结束 | |

图 10-13 选择模板

步骤5 进入"任务编排"页面后,单击"新建阶段"。单击 2 将"阶段_1"改名为"构建","阶段_2"改名为"部署"。

图 10-14 新建阶段

| ← | software_release_libr | ary_pipe | . 🖻 | 基本信息 | 任务编排 | 参数设置 | 执行计 | 划 杉 | 又限管理 | 通知订 | 阅 | | | | | | |
|---------------|-----------------------|----------|---------------------|-----------------|---------------------|------|--|-----------------|------|-----|-----|---|--|-----|----|--|--|
| 济 1 | 記水线源 个代码源 | Ū | 构 1~ | 建 ↑任务 | | 0 | · | 署 N任务 | | Ø | 0 0 | J | | | | | |
| | 🗘 Repo | •—€ | 0 | ③ 新建任約 | ጅ + | -0 | ••••• | | + 新翅 | 任务 | | • | | ⊢新建 | 阶段 | | |
| | cae-backend | | | | | | | | | | | | | | | | |
| | ₽ master | | | · · · · · | 4000 | | | | ピ准出 | 1条件 | | | | | | | |
| | 别名: | | | 1 | + 开门士劳 | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | |
| | | | | | 准出条件 | | | | | | | | | | | | |
| | | | | | - The second second | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | |

步骤6 新建构建任务。

- 在"构建"阶段,单击"新建任务"。
 - a. 进入"新建任务"页面,单击"构建",选择"Build构建"任务,并单击 "添加"。

图 10-15 新建任务

新建任务 🧷

| ← software_release_library_pipel □ | 基本信息 任务编排 参数 | 改设置 执行计划 权 | 限管理 通知订阅 | |
|------------------------------------|---|--------------------------|----------|--------|
| 流水线源 1 个代码源 □ 1 | 内建 の 日本 の 日 | 回 III 部署 0 个任务 | 2 O D | |
| ✓ Repo | ⑦ 新建任务 + | -••••© | + 新建任务 | + 新建阶段 |
| 译 master 别名: | + 并行任务 | | じ 准出条件 | |
| | ピ 准出条件 | | | |

图 10-16 添加 "Build 构建"任务

| 请从右侧选择插件 | Q、请输入关键字 | 👌 更多插件 |
|----------|---|--------|
| | 全部 内科理 代码检查 部署 测试 通用 | |
| | Build构建 ۞ 任务 调用CodeArts Build能力进行构建。编译构建服务(CodeArts Build)为开发者提… | 添加 |

b. 选择前一步创建的"编译构建任务"和"仓库",将"release_version"设 置为流水线参数,单击"确定"。

ΰX

图 10-17 bucket_name

| Build构建 🧷 | | ī | | × |
|-------------------|---|------------|----------|--------|
| 任务配置 Build构建 : | ● Build构建 ● 调用CodeArts Build能力进行构建。编译构建服务(CodeArts Build)为开发者提供配置简单 ・各称 | ① 温 É 屠 | 馨排 夏开 | 是示 |
| | Build构建 ·请选择需要调用的任务 ② 找不到合适的任务?点击创新 cae-backend-obs | 建 | 刷師 |) 新 |
| | 参数名称: bucket_name 设置为流水线参数,支持任务中引用该参数 设置为流水线参数 | | • | |
| | file_name cae-backend jar | | | |

图 10-18 file_name

| Build构建 🧷 | | ▣ | \times |
|-------------------|--|--------------|--|
| 任务配置 Build构建 : | ● Build构建 ◎ ① 调用CodeArts Build能力进行构建。编译构建服务(CodeArts Build)为开发者提供配置简单。 ・ 名称 Build构建 |)温馨: - 展开 | 提示 F |
| | 请选择需要调用的任务 ② 技不到合适的任务?点击创建 cae-backend-obs 仓库 cae-backend | 刷 | st in the second |
| | 参数名称 file_name 设置为流水线参数,支持任务中引用读参数 设置为流水线参数 Cae-backend.jar | | |

- 在"部署"阶段,单击"新建任务"。 •
 - a. 在搜索框中输入"CAE",并单击 🔍 。选择"CAE发布"插件,单击"添加"。

图 10-19 选择插件

| ← software_release_lib | wary_pipel II 副本語語 《《例論》 ##202月 #2月11名 #2822月 #2027月 | 新聞 王術 // | × |
|---|---|---|----------------|
| 波水线源 | B ⊨ <mark>1907</mark> // 0 B ⊨ <mark>8990</mark> // 0 B | WK.Edeb35869 | ▲ Ⅱ乐福件 |
| CÓ Repo | | <u>\$10</u> nat (1522) 1549 1955 369 | |
| * () cae-backend * 2² master * 89%: | - RGER ØRURA | CAE线在 • CAE (Cloud Application Engine完成明) 是一个最短度用的的问题 | 28.20 |
| | G mann | CAE发布 @ CAE X AE X AE X AE X AE X AE X AE X | 张振兴,道 。 |
| | | | |

b. 配置插件参数后,单击"确定"。

= 🐵 💼 🗙

- 区域选择:待部署的区域,与当前CodeArts在同一个Region。
- 环境:选择组件所属环境。
- 应用:选择组件所属应用。
- 组件:选择待升级的组件(需要选择在CAE中使用软件包部署的组件)。
- 组件来源:选择"软件包"。
- 运行时:选择软件包对应的运行时。
- 软件包仓库:选择"CodeArts Artifact"。
- 软件包地址:取值见参数说明。

图 10-20 配置插件参数

CAE发布 🖉

| CAE发布 : | CAE发布 🍨 V1.0.0 👻 CAE (Cloud Application Engine云应用引擎) 是一个面向应用的Serverless托管服 | ① 温馨 务,提供… 展 3 |
|---------|---|--------------------------|
| 添加插件 | *名称 | |
| | CAE发布 | |
| | *区域选择 | |
| | | |
| | *环境 | |
| | | |
| | *应用 | |
| | | |
| | *组件 | |
| | | |
| | *组件来源 | |
| | ○ 镜像 ● 软件包 | |
| | *运行时 | |
| | Java8 | |
| | 软件包仓库 | |
| | OBS CodeArts Artifact | |
| | 软件包地址 | |
| | https://devrepo. 3/download?fi | ilename=897e |

步骤7 单击"保存并执行",进入执行配置页面,确认执行配置信息无误后,单击"执行"。

图 10-21 执行任务

| software release library pipel D NORE COMMON PRODUCT BUTHER DISTINGTING COMMON AND A COMMON PRODUCT BUTHER DISTINGTING COMMON AND A COMMON PRODUCT BUTHER DISTINGT AND A COMMON PROD | | 0 114 94 647161 5 |
|--|---------------|--------------------------|
| A rese C ore school D ore school D ore school C maar C ma | → + man n | |
| 图 10-22 确认执行配置信息 | | |
| 执行配置 | | × |
| 流水线源 | | |
| 仓库 | 分支/标签 | |
| 🏈 cae-backend | 分支 ▼ master | • |
| ▼ 运行参数配置 | | |
| 名称 | 值 | |
| release_version | \${TIMESTAMP} | |
| ▼ 执行阶段配置 | | |
| 🖃 🗹 所有阶段 | | |
| - 🖃 💙 构建 | | |
| CloudBuild | | |
| - 🛃 部署 | | |
| ✓ CAE发布 | | |
| | | |

步骤8 查看流水线执行结果。

图 10-23 查看流水线执行结果

| 流水线详情 | 执行历史 | | | | | | | |
|-----------------------------------|----------------------------|---|---------------------|--------------------------------|-------|--------------------|-------------|-------|
| 流水线源 1 个代码源 | | 0 | 构建 1/1个任务 | 务完成 3m12s | ø | 部署 1/1个任 | 务完成 1m10s | 0 |
| ✔ Rep ☆ cae-1 浆 mast 别名: | po backend er | - | -@ | CloudBuild | 3m12s | -@- | SCAE发布 | 1m10s |

步骤9 登录CAE控制台,在组件列表查看组件状态,"最近一次变更状态/时间"列显示状态为"升级成功",则表示Codearts流水线执行升级成功。

图 10-24 查看组件状态

| ◎ 云应用引擎 | ₩ 环境: • test • | · ① ⑥ 应用: 1 | est 🔻 🤆 | € | | | | | |
|--------------|----------------|-------------------------|---------|------------|--------------------------------------|--|-------------------------------|-----------|---------|
| 概范 始件列表 | 新聞組件 | | | | | | | 春輸入名称进行撤卖 | Q C 🐵 |
| 实例列表 | 名称/版本 ⇔ | 代码源 🍞 | 秋恋 🍞 | 实例个数 (可用)全 | 访问地址 | 最近一次变更状态时间 ⇔ | 创建时间 🖕 | 快捷跳转 | 操作 |
| 組件配置 组件事件 | test v1.0.1 | 软件包 cae-backend.jar | ① 未就績 | 1/1 | | ● 升级成功 Q 2024/01/15 14:59:05 GMT+08:00 | 2024/01/15 14:56:55 GMT+08:00 | 6066 | 升级 更多 ▼ |
| 细件监控 组件日志 | #ee v1.00 | 読録 nginx stable-perl | 3 运行中 | 1/1 | 87 | ● 部署成功 [Q 2024/01/09 20:34:56 GMT+08:00 | 2024/01/09 20:34:55 GMT+08:00 | 6066 | 升级 更多 ▼ |
| 系统设置 | 90tt v1.0.5 | 讀像 nginx stable-perl | 3 运行中 | 1/1 | ☐ http://110.238.106.215.9966 (公网地址) | ● 升级成功 [Q 2024/01/09 20:35:02 GMT+08:00 | 2024/01/08 19:44:16 GMT+08:00 | 6066 | 升级 更多 ▼ |
| | | | | | | | | | |

----结束

参数说明

| 参数名称 | 是否必须 | 参数类型 | 描述 |
|---------------------|------|--------|---|
| release_ver sion | 是 | String | 发布版本号,取值使用时间戳变 量:\${TIMESTAMP}。 |
| 软件包地址 | 是 | String | 编译构建出来的软件包上传到软件 发布库中的地址。 例如:复制 图10-9 中下载地址,下 载地址为https:// devrepo.devcloud.cn- east-3.huawei.com/ DevRepoServer/v1/files/ download? filename=39d25a3cc6ee48678533 020abcfbf941/cae-backend/ 20231028200733/cae- backend.jar,将软件包的目录发布 版本号20231028200733改为\$ {release_version},则软件包地址 的值为https:// devrepo.devcloud.cn- east-3.huawei.com/ DevRepoServer/v1/files/ download? filename=39d25a3cc6ee48678533 020abcfbf941/cae-backend/\$ {release_version}/cae- backend.jar。 |

表 10-1 参数说明

10.3 流水线构建软件包上传到 obs 桶后升级 CAE 组件

创建编译构建任务

创建代码仓库的构建任务:

- 步骤1 进入CodeArts控制台,单击右上角"立即使用"。
- 步骤2 在菜单栏选择"服务 > 编译构建"。

🗱 🌺 华为云 控制台 💧 💼 首页 膺 工作台 👼 看板 🥥 服务 • 📕 华为开源镜像站 4 😪 💿 **S**New ☆ 我的关注 C/S </> 1 Hello, 🗊 所有项目 代码托管 流水线 代码检查 IDE 编译构建 高 已归档 您还没有项目,从创建一个 . 27 8 ×. 部署 扩展插件 制品仓库 测试计划 移动应用测试 项目模板 **(**) 0 1 3 1 文档 Wiki DevStar 7 APIOps 漏洞管理ス 0 IPD-系统设备类 🎑 开源治理》

步骤3 在"编译构建"页面单击"新建任务",在基本信息中填入"任务名称",并选择 "归属项目"、"源码源"、"源码仓库"和"分支",单击"下一步"。

图 10-25 选择服务

| | | | ⇒* 亲 | 新建构建任务 | | | |
|------|-----------------|--------|-----------------------|-------------------|----------------|---|--|
| 基本信息 | | | | | | | |
| 构建模板 | 基本信息 | | | | | | |
| | •任务名称 | | | | | | |
| | cae-backend-obs | | | | | | |
| | * 归属项目 | | | | | | |
| | cae-teset | | | | v | | |
| | * 源码源 | | | | | | |
| | CÓ Repo | GitHub | ● 通用Git | GitCode | 日 码云 | | |
| | GitLab | Gerrit | () 其他项目Repo | 《 来目流水线 | | | |
| | *源码仓库 | | | | | | |
| | cae-backend | | | | Ŧ | | |
| | *分支 | | | | | | |
| | master | | | | | | |
| | 任务描述 | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | 0.542 | | |
| | | | | | 0/512 | 2 | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

步骤4 选择构建使用的模板(此示例使用的为java代码,故此步选择Maven模板)后,单击"下一步"。



图 10-26 配置构建基本信息

| | | | | 💕 新建构建任务 | 5 | |
|--------|----------|----|-------------|--|---------------------|---|
| ✓ 基本信息 | **** | | 18-22-17-12 | | | |
| 🔗 构建模板 | 竹建快饭 | | 1日191实权 | | 1 | |
| | Q 搜索模板名称 | | Maven | Maven 使用Apache Maven构建Java项目 | | 0 |
| | 凸 推荐模板 | 1 | | | | |
| | 郘 公共模板 | 37 | 公共模板 | | | |
| | A。自定义模板 | 0 | | 空白构建模板 使用系统构建步骤自由定义构建任务。 | | |
| | | | * | Service Stage-Maven-镜像构建 使用Apache Maven构建Java项目,通过Dock | erfile制作镜像并推送到SWR仓库 | |
| | | | * | ServiceStage-Maven-jar包构建 使用Apache Maven构建Java项目,打成Jar包 | 上传到没布库 | |

步骤5 进入"构建步骤"页面后,选择"添加步骤",单击"文件上传",选择"上传文件到OBS",单击"添加"。

图 10-28 文件上传

| ← cae-backend-obs 基本項目 内職步費 学数位置 执行対対 | |
|--|--|
| 2010 (1959): 4005740520 400540652 | 添加片板 系称手板 直面形板 <u>又和上時</u> 田田林 Q 2010人70271020 |
| CLID TEMACIN REIN-FRI TRADIC | 新情報會升超送到SWR仓库 通正Doutentelennee并用述ESONADa、 書簡值的指面者 |
| Marcer Marcerigille | ■ 上作能作也到现代形式也成 上考虑中心的2000年末期回题 |
| L-RECH 65100112 (A.6. | LiftXffMoes |
| | |

- 步骤6 进入"参数设置"页面,参考<mark>图10-29</mark>配置构建参数和默认值,并开启运行时设置。 bucket_name:上传到obs的桶名称,如caetest。
 - file_name: 上传到obs桶中的软件包名称, 如cae-backend.jar。

图 10-29 配置构建参数

| ← cae-backend-obs 基本信息 内健分数 | 进行计划 | | | | |
|---|------|-----------------|--------|---------------------|-------------|
| 日主文参数 系统预建文参数 Q、INA入共留字 | | | | | |
| 8 0 | 黄檗 | 默认值 | 私業作政 🛞 | 18179922 2 🛞 | 希数描述 |
| codeBranch | 字符串 | master | | | 代码分支系统预定文参数 |
| bucket_name | 字符串 | caetest | | | |
| file_name | 字符串 | cae-backend jar | | | |
| + 新建佛教 | | | | | |

步骤7 进入"构建步骤"页面,删除"上传软件包到软件发布库"步骤。

| | ← cae-backend-obs 基本信息 校識步骤 参数设置 执行计划 修改历史 权限管理 通知 | |
|--|--|--|
| 代野下監定課 上序校计电频软件技术由率 取用代码下电方式 * 代題を出発記 ① ** 代題を出発記 ② ************************************ | 開発化 代码化 株園野地和2000 東国市地和2504金田田田町町400 | ■ 上传软件包到软件发布库 上传软件包到软件发布库 置有操作指向 ・步骤显示态称 |
| | ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ | 上传软件数据软件数例本 * foliaba38经 ④ **targetr:?ar |
| Moven (Bill Streets) (Breache Movent) (B | Mover Without Wavestill | 发布版本号 🔘 |
| | | 12A 0 |
| Ethodal State | LHCXHNOBS 上HCXHNOBS. 3 新用 | |

图 10-30 删除"上传软件包到软件发布库"步骤

步骤8 在弹框中单击"确定",完成删除。

- 步骤9 配置"上传文件到OBS"参数。
 - "配置构建产物路径"为"**/target/*.?ar"。
 - "桶名"使用参数"\${bucket_name}"。
 - "OBS存储目录"使用默认配置。
 - "OBS存储文件名"使用参数"\${file_name}"。

图 10-31 配置"上传文件到 OBS"参数

| ← cae-backend-obs 基本信息 构建步数 参数设置 执行计划 修改历史 权限管理 通知 | | | |
|---|---|---|----------|
| ← cae-backend-obs ##### mittel® entropy ###### entropy #################################### | | LR2x#ploss Lr2x#ploss Lr2x#ploss Lr2x#ploss More Ga Mo | |
| Acceleration | • | [""sgart the " #6 0 Costraget 0 / Costraget 0 / Costraget 0 BR_ithOrthe 0 & 0 Costraget 0 Costrag | |
| | | बर ब (१९९८), तक (१७) | 1819 |

步骤10 单击右上角"新建并执行",确认运行时参数无误后,单击"确定"。

| | | | | 取消 | 新建新建并执行 |
|----|------------------------|-----------------|---|----|---------|
| | | | | | |
| | | | | | |
| ŧ | 丸 行 | | × | | |
| jā | 运行时参数 ^{名称} | 值 | | | |
| | bucket_name | caetest | | | |
| | file_name | cae-backend.jar | | | |
| | 确定 | 取消 | | | |
| | | | | | |

图 10-32 执行任务

步骤11 查看构建日志,等待构建成功并成功上传软件包到OBS中。



图 10-33 查看构建日志

步骤12 登录OBS控制台,在OBS桶中查看软件包已上传成功。

图 10-34 查看 OBS 桶

| 对象 | 已删除对象 | 碎片 | | | | |
|-------|---------------------------|-------------|----------------------------|-----------|------------|------------|
| 若需要将对 | 「象移动到桶内其他位」 「象」 新建文 | 置,推荐下载使用OBS | 6 Browser+图形化管理工具。 更多 ~ | 基于安全合规要求, | 从浏览器直接访问文件 | 时不能进行在线预览; |
| 5 | 2、輸入对象名前缀搜 | 索 | | | | |
| 日本 | ¥ | | 存储类别 | | | 大小 ⇔ |
| | cae-backend.jar | | 标准存储 | | | 55.18 MB |

----结束

创建流水线使用 OBS 软件包升级 CAE 组件

创建Codearts流水线:

- 步骤1 返回Codearts控制台首页。
- **步骤2** 在菜单栏选择"服务 > 流水线",在"流水线"页面单击"新建流水线"。

图 10-35 选择服务

| ₩ 华为云 | 控制台 | a 🗴 🔹 🔹 👌 首页 🚔 工作台 💆 管理看板 | ◎ 服务 ▼ | | | | |
|--------------|--------|--------------------------------|----------|------------|-------------|-----------|----------|
| cae-teset | | 首页 / cae-teset / 流水鏡 | | | Г | | |
| T./m | | 流水线 策略 | - 22 | \$ | | 1 | |
| 山TF 初時時間 | | 三 ■ 全部分組 + 新設施水线 全部流水线 ▼ Q 清輸入 | 架构建模 | 代码托管 | 代码检查 | 流水线 | 编译构建 |
| 2451-924E146 | | ○ 名称 最近一次 | | | | | |
| 1709 | , | #26 cae | * | 1 | | | 3 |
| 持续交付 | \sim | Cae-demo 12 maste | 测试计划 | 部署 | 制品仓库 | Wiki | 文档 |
| 流水线 | | | | | | | |
| 发布管理 | | | ۷ | - | | \sim | 1 |
| 编译构建 | | | 扩展插件 | IDE Online | IDE | DevStar > | APIOps 7 |
| 部署 | | | | | | | |
| 运维 | | | (| 0 | Flow | | |
| 制品仓库 | > | | 漏洞管理服务》 | 开源治理服务ス | Astro 工作流 > | | |
| 测出式 | > | | | | | | |
| | | | | | | | |
| 图 10-3 | 86 亲 | 所建流水线 | | | | | |

| 🌛 流水线 | | | |
|---|-----------------------------|-----|--|
| 流水线列表 模板列表 | | | |
| +新建流水线 ► Q 请输入员 | 冰线名称 | | |
| 名称 👅 | 最近一次执行 | 工作流 | |
| NewPipeline-20240115143126 codearts-test-30050199 | #2 cae-backend #3 master | 0-0 | |

步骤3 在"新建流水线"界面,输入"名称"并选择"流水线源"、"代码仓"和"默认分支",单击"下一步"。

| 图 10-37 | 配置流水线基本信息 |
|---------|-----------|
|---------|-----------|

| •乐菜四日 Car-test OS_pipeline •无状结束 CitHub EFSC#C ERPO CitHub EFSC#C ERPO CitHub EFSC#C EFSC#EFSC#C EFSC#EFSC#C EFSC#EFSC#EFSC#EFSC#EFSC#EFSC#EFSC#EFSC# | ^{体信息} 基本信息 | | | | | | | |
|--|---------------------|--------------------------------|--------|---------|--|--|--|--|
| cae-test • •Cristi • obs_pipeline • •Cristi • ●*Cristi ● ●*Cristi ● ●*Cristi ● ●< | *所属项目 | | | | | | | |
| ・伝統 ● た成状 ● 作成 | | | | - | | | | |
| obs_pipeline *Exytdge Pepo GitHub Withing Withing Withing Withing BHGIt GitCode BHGIt Withing Cae-backend Withings State BHGIt | *名称 | | | | | | | |
| ・KGAKBEE ● C ● C ● C ● C ● C ● B ● C ● B | obs_pipeline | | | | | | | |
| Image: Section of the section of t | *流水线源 | | | | | | | |
| Repo GitHub 研云Git Gerrit GitLab ● | <u>م</u> | 0 6 | 1 | ₩ | | | | |
| Weight Geo 通用Git 画面 *代码合 週用Git 通用Git 第不通序 *代码合 期所 画面 ● *低码合 ● 第所 ● *低日 ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● <td>Repo</td> <td>GitHub 码云Git</td> <td>Gerrit</td> <td>GitLab</td> <td></td> <td></td> | Repo | GitHub 码云Git | Gerrit | GitLab | | | | |
| TFS改库 通用Git GitCode 通用Git 暫不选择 *代码会 勝所 Cae-backend ・ *默认分支 ● master ・ Repo https提校 ● 新建版校 通道译 ・ 別名 ● 別名仅近時輸入大小码先文字母、数字、'', 最多128个字符 描述 语項唱比大线删述 | O ni | G G G | • | | | | | |
| cae-backend *就以分支 ◎ master Repo https提役 ◎ 新建成权 请选择 ● 助名 ◎ ● 励名 ② ● 加速 ● 加速 ● 前如可此水线照送 ● | TFS仓库 *代码仓 | 通用Git GitCode | 通用Git | 智不选择 刷新 | | | | |
| *飲以分支 ◎ master Repo http:提校 ◎ 新建版仪 请选择 第名仪过36歳入大小写完文字母、数字、'_', 最多128个字符 新成 描述 请编写论水线描述 | cae-backend | | | • | | | | |
| master ● Repo https接役 ● 新建設び 博造塔 ● 第名 ● ■ 副名仅近時輸入大小写完文字母、数字、'_'、最多128个字符 ■ 描述 ■ 请编写论大线描述 ● | *默认分支 🔘 | | | | | | | |
| Repo 新建模型 通過塔 第名 @ 期名仪支持输入大小写美文字母、数字、'_', 最多128个字符 描述 請項写助水线描述 | master | | | - | | | | |
| ● 通志择 ● 射名(②) 卵名(文法移輸入犬小写英文字母、数字、'_', 量多128个字符 描述 請項写论大线照差 | Repo https授权 🔘 | | | 新建授权 | | | | |
| 第名 ● 第名反支持输入大小同先文字母、数字、'_, 最多128个字符 運送 環境可能水线描述 | 请选择 | | | ~ | | | | |
| 财名仪过芬输入大小写死文字母、数字、'_',最多128个字符 描述 请項写说水线描述 | 別名 🔕 | 別名 💿 | | | | | | |
| 描述 读描写论水线描述 | 别名仅支持输入大小 | 别名仪支持输入大小写英文字母、数字、'_',最多128个字符 | | | | | | |
| 诺斯写论火线描述 | 描述 | | | | | | | |
| | 请填写流水线描述 | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |

步骤4模板选择"无>空模板",单击"确定"。

| 模板 | 模板选择 | 自定义 | 模板 | 記 新建模板 | ः≣ 收起 |
|----|---------|-------|----------------------------------|---------------|-------|
| | Q搜索模板 | | template-20211026153921 | | |
| | 88 全部 | 20 | | | |
| | 🛓 Java | 8 | | | |
| | Node.js | 5 | ServiceStage-Maven-虚机部署-copy | | |
| | Python | 2 | 开始(代码检查) 构建 | 曙 结束 | |
| | ∞ Go | 1 系统模 | 扳 | | |
| | ● PHP | 1 | 空模板 | | 0 |
| | ↔ 其他 | 1 | 开始 结束 | | |
| | @ 无 | (2) | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

🌛 新建流水线

步骤5 进入"任务编排"页面后,单击"新建阶段"。单击 2 将"阶段_1"改名为"构建","阶段_2"改名为"部署"。

图 10-39 新建阶段

图 10-38 选择模板

| 流 7 1 个 | 火线源 一代码源 | Û | ■ 构建 0 个 | | □ #書 0 个任 | 〕 务 | 1 | | |
|-------------------|--------------------|------|-------------|--------|--------------|---------------|-------------|--------|--|
| | 🐼 Repo | •——• | 0 | + 新建任务 | | + 新建任务 | - ⊕- | + 新建阶段 | |
| | 🗇 cae-backend | | | | | | | | |
| | ₽ master | | | ℃ 准出条件 | | じ 准出条件 | | | |
| | 别名: | | | | | | | | |
| | | | | | | | | | |

步骤6新建构建任务。

- 在"构建"阶段,单击"新建任务"。
 - a. 进入"新建任务"页面,单击"构建",选择"Build构建"任务,并单击 "添加"。

•

图 10-40 新建任务

| 流水线源 1 个代码源 | 3 | / 6 |) 🗇 | | |
|----------------|----------|------------------|-----|------------|--|
| | | | | | |
| | 上的地位在 | 1. ++2+2+2/17/42 | | 1. 辛运动的 60 | |
| 🗘 🐼 Repo | 〒 新经1155 | 〒 初建江另 | | 〒 朝廷明叔 | |
| 🗈 cae-backend | | | | | |
| f master | ピ 准出条件 | じ 准出条件 | | | |
| * 别名: | | | | | |
| | | | | | |
| | | | | | |

图 10-41 添加 "Build 构建"任务

| 新建任务 🧷 | | ۵ × |
|----------|---|--------|
| 请从右侧选择插件 | Q 请输入关键字 全部 <u>构建</u> 代码检查 部署 <u></u> | ▶ 更多插件 |
| | Build构建 ③ 任务 调用CodeArts Build能力进行构建。编译构建服务(CodeArts Build)为开发者提… | 添加 |

b. 选择前一步创建的"编译构建任务"和"仓库",并将创建编译构建任务<mark>步</mark> <mark>骤6</mark>配置的"bucket_name"和"file_name"设置为流水线参数,单击"确 定"。

图 10-42 bucket_name

| Build构建 🧷 | 10 10 | ĭ× |
|-------------------|---|----------|
| 任务配置 Build构建 : | Build构建 の ① 温 週間CodeArts Build能力进行构建。编译构建服务(CodeArts Build)为开发者提供配置简单… 展 ・名称 | 馨提示 开 |
| | Build构建 • 请选择需要调用的任务 ② 找不到合适的任务?点击创建 」 | 刷新 |
| | cae-backend-obs | • |
| | 参数名称: bucket_name 设置为流水线参数, 支持任务中引用该参数 设置为流水线参数 | • |
| | * file_name cae-backend jar | |

图 10-43 file_name

| Build构建 🧷 | | 1 | \times |
|-------------------|--|-------------|----------|
| | | | |
| 任务配置 Build构建 : | Build构建 ・ 名称 ・ 名称 ・ 名称 |) 温馨: 展开 | 提示 |
| | Build构建 | | |
| | 请选择需要调用的任务 ◎ 找不到合适的任务?点击创建 cae-backend-obs | 刷 | 新 |
| | * 仓库 | | |
| | cae-backend | | ٢ |
| | 参数名称: file_name 设置为流水线参数, 支持任务中引用该参数 <mark>设置为流水线参数</mark> Cae-backend.jar | | |

- 在"部署"阶段,单击"新建任务"。
 - a. 在搜索框中输入"CAE",并单击 🔍 。选择"CAE发布"插件,单击"添加"。

图 10-44 添加部署任务

| 基本信息 任务编排 参数设置 执行计划 权限管理 通知订阅 | 新進任务 🖉 × |
|---|---|
| Number Image: Transmission of the state of | 演从右側地呼腦件 Q ca2 ④ 更多插件 全部 仲操 代码检查 部項 則此 通用 |
| ・ + #行任务 の湿出条件 の湿出条件 | 3 CAE (Cloud Application Engine完約用別等) 第一个银标题用的Servertes時間… Bab |

- b. 配置插件参数后,单击"确定"。
 - 区域选择:待部署的区域,与当前CodeArts在同一个Region。
 - 环境:选择组件所属环境。
 - 应用:选择组件所属应用。
 - 组件:选择待升级的组件(需要选择在CAE中使用软件包部署的组件)。
 - 组件来源:选择"软件包"。
 - 运行时:选择软件包对应的运行时。
 - 软件包仓库:选择"OBS"。
 - \$\$\$\$ 软件包地址:取值见参数说明。

| CAE发布 ⊘ | | ð × |
|---------|--|-----|
| САЕ发布 : | CAE发布 ♥ V1.0.0 ▼ ① 温 CAE (Cloud Application Engine云应用引擎) 是一个面向应用的Serverless托管服务, 提供… 雇 | 馨提示 |
| 添加插件 | *名称 | |
| | CAE发布 | |
| | *区域选择 | |
| | | • |
| | *环境 | |
| | | * |
| | *应用 | |
| | | • |
| | *组件 | |
| | | - |
| | *组件来源 | |
| | ○ 镜像 ● 软件包 | |
| | *运行时 | |
| | Java8 | - |
| | 软件包仓库 ● OBS OcodeArts Artifact | |
| | 软件包地址 | |
| | https://\${bucket_name}.\${obs_address}\${file_dir}\${file_name} | |

图 10-45 配置插件参数

- c. 进入"参数设置"页面,参考<mark>参数说明</mark>添加自定义参数。
 - bucket_name: obs桶名称, 如caetest。
 - file_name: 上传到obs桶中的软件包名称, 如cae-backend.jar。
 - file_dir:软件包上传到obs桶中的存放目录,如根目录是"/",根目录 下面的test目录"/test/"。
 - obs_address:为OBS桶的地址,名称格式为:obs.区域项目名称.myhuaweicloud.com。

图 10-46 配置自定义参数

| ← □ 基本信息 任务编排 | 参数设置 执行计划 | 权限管理 通知订阅 | | | |
|---------------------------|-----------|-----------------------|--------|---------|------|
| 自定义参数 系统预定义参数 Q 请给人名称或默认值 | | | | | |
| 名称 | 类型 | 默认值 | 私憲參数 🛞 | 运行时设置 🛞 | 参数描述 |
| file_dir | 字符串 | 1 | | | |
| obs_address | 字符串 | obs. yhuaweicloud.com | | | |
| bucket_name | 字符串 | | | | |
| file_name | 字符串 | cae-backend.jar | | | |
| 十 新建参数 | | | | | |

步骤7 单击"保存并执行",进入执行配置页面,确认执行配置信息无误后,单击"执行"。
图 10-47执行任务

| ← obs_pipeline 🖻 🐰 | 本信思 f | 王劳纳旧 | 李敬说置 执行计划 6 | 用管理 通知订 | я | | | | | | | | | ۲ | 100H | 保持 | 保存开执行 |
|-------------------------|--------------|---------------------|-------------------|--------------|---------------|---|--------|--|--|--|--|--|--|---|------|----|-------|
| 流水线罩 1 个代码即 | 1 | 日 将 到 1 个 | 1 (19) | 8 I 8 | | 8 | | | | | | | | | | | |
| Repo Constructioned | • | 0 | ⊗ CloudBuild + | • • | & CALZER + | | + 新建阶段 | | | | | | | | | | |
| P master 副帝: | | | + 州行任务 | | + #行任务 | | | | | | | | | | | | |
| | | | ul 准出条件 | | 达 潮出条件 | | | | | | | | | | | | |

图 10-48 确认流水线配置信息

流水线源

| 仓库 | 分支/标签 | | | | | | |
|---------------|-------|--------|---|--|--|--|--|
| 🐼 cae-backend | 分支 🔻 | master | - | | | | |

▼ 运行参数配置

| 名称 | 值 | | | | |
|-------------|-----------------|--|--|--|--|
| bucket_name | caetest | | | | |
| file_name | cae-backend.jar | | | | |
| file_dir | 1 | | | | |
| obs_address | obs cloud.com | | | | |

▼ 执行阶段配置

| 🖃 🚽 所有阶段 |
|--------------|
| - 🗸 构建 |
| 🖂 CloudBuild |
| - 🖂 部署 |
| ✓ CAE发布 |

步骤8 自动跳转至流水线详情页面,查看流水线执行结果。

图 10-49 查看流水线执行结果



步骤9 登录CAE控制台,在组件列表查看组件状态,"最近一次变更状态/时间"列显示状态为"升级成功",则表示Codearts流水线执行升级成功。

新增组件 名称/版本 ≜ 代码源 🍞 实例个数 (可用/全... 访问地址 最近一次变更状态/时间 👙 状态 🍞 Java test-obs v1.0.1 软件包 ● 升级成功 🔽 ● 运行中 1/1 🖉 cae-backend.jar 2024/01/24 14:06:36 GMT+08:00 ffee v1.0.0 镜像 nginx stable-perl 停止成功 已停止 0/1 2024/01/24 11:12:00 GMT+08:00

----结束

参数说明

表 10-2 参数说明

图 10-50 查看组件状态

| 参数名称 | 是否必须 | 参数类型 | 描述 |
|-----------------|------|--------|--|
| bucket_na me | 是 | String | obs桶名称,CAE发布插件中选择的区域 和obs桶所在的区域保持一致。 |
| file_name | 是 | String | 上传到obs桶中的软件包名称。 |
| file_dir | 是 | String | 软件包上传到obs桶中的存放目录,目录 需要以/结尾,如果obs桶中没有这个目 录,会自动创建出该目录。如根目录是 "/",根目录下面的test目录"/ test/"。 |
| obs_addres s | 是 | String | 值的格式为:obs.区域项目名 称.myhuaweicloud.com,区域项目名称 就是region,参考 <mark>参数值获取</mark> 。 |
| 软件包地址 | 是 | String | 上传到obs的软件包地址,值填写: https://\${bucket_name}.\${obs_address} \${file_dir}\${file_name}。 |

10.4 流水线构建镜像上传到 swr 镜像仓库后升级 CAE 组件

创建编译构建任务

创建代码仓库的构建任务

- 步骤1 进入CodeArts控制台,单击右上角"立即使用"。
- 步骤2 在菜单栏选择"服务 > 编译构建"。

| 🔛 🌺 华为云 控制台 | 💧 首页 🚔 工作台 💆 看板 | 👌 服务 ▼ 📃 | 华为开源镜像站 | | | ¢° ⊘ ¢ |
|-------------|-----------------|----------|---------|-----------|----------|--------|
| ☆ 我的关注 | 11-11- | c/s | | ¢ New | 3 | |
| 前有项目 | Hello, | 代码托管 | 代码检查 | IDE | 流水线 | 编译构建 |
| а 已归档 | 您还没有项目,从创建一~ | ì | | | | |
| | | - | ۸ | 6 | | - |
| | 项目模板 | 部暋 | 扩展插件 | 制品仓库 | 测试计划 | 移动应用测试 |
| | RR FE | 3 | | 522 | 1 | 0 |
| | SR D | 文档 | Wiki | DevStar ≫ | APIOps 🤊 | 漏洞管理ス |
| | AR | 0 | | | | |
| | IPD-系统设备类 💷 🛛 | 开源治理 >> | | | | |

图 10-51 选择服务 ::: 🎎 #为云 控制台

步骤3 在编译构建页面单击"新建任务",在基本信息中填入"任务名称",并选择"归属项目"、"源码源"、"源码仓库"和"分支",单击"下一步"。

图 10-52 新建构建任务

| 编译构建 成功率 91% 任务个数 4 | | |
|---------------------------------|----------|----------|
| + 新建任务 Q 搜索任务名称 | | |
| 名称 👅 | 最近一次执行 💿 | 最近执行结果 |
| Codearts-test / cae-backend-obs | | 11111111 |

| | | | 💕 亲 | 新建构建任务 | | |
|--------|-----------------|--------|----------------|-------------------|---------|--|
| ∂ 基本信息 | | | | | | |
| 🕤 构建模板 | 基本信息 | | | | | |
| | *任务名称 | | | | | |
| | cae-backend-swr | | | | | |
| | •归属项目 | | | | | |
| | cae-teset | | | | Ŧ | |
| | * 源码源 | | | | | |
| | Repo | GitHub | o 通用Git | GitCode | G 码云 | |
| | GitLab | Gerrit | 其他项目Repo | 《 来自流水线 | | |
| | *源码仓库 | | | | | |
| | cae-backend | | | | Ŧ | |
| | *分支 | | | | | |
| | master | | | | | |
| | 任务描述 | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | 0/512 | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

步骤4 选择构建使用的模板(此示例使用的为java代码,故此步选择Maven模板)后,单击"下一步"。

图 10-54 选择构建模板

图 10-53 配置构建任务基本信息

| 🥑 基本信息 | | | | | | |
|--------|----------|----|-------|--|---------------------|---|
| 🕖 构建模板 | 构建模板 | | 推荐模板 | | 1 | |
| | Q 搜索模板名称 | | Maven | Maven 使用Apache Maven构建。Java项目 | | 0 |
| | 查 推荐模板 | 1 | | PO Di Branci marchi all'Estandi Chi | | |
| | 88 公共模板 | 37 | 公共模板 | | | |
| | ♀。自定义模板 | 0 | | 空白构建模板 使用系统构建步骤自由定义构建任务。 | | |
| | | | * | Service Stage-Maven-镜像构建 使用Apache Maven构建Java项目,通过Docke | erfile制作鏡像并推送到SWR仓库 | |
| | | | | ServiceStage-Maven-jar包构建 | | |
| | | | ٠ | ServiceStage-Maven-jar包构建 | 一位到发布库 | |

步骤5 进入"构建步骤"页面后,删除"上传软件包到软件发布库"步骤。

图 10-55 删除"上传软件包到软件发布库"步骤

| ← cae-backend-swr 基本信息 构建的 | · 骤 参数设置 执行计划 | |
|----------------------------|---|--|
| | 图形化 代码化 | ► 上传软件包到软件发布库 上传软件包到软件发布库 查看展作指离 |
| | 构建环境配置 成置构建任务全局运行环境 | 1 LINE 2.2 C |
| | ↓. 代码下线配置 配置代码下约方式 | と MARIAN CALVER からしていたいです。 上代ない件の知知な件没有成本 を必須知知ないでの。 |
| | Mavenikit# | evisitione: |
| | maven 使用Apache Maven构建Java项目。 | 发布版本号 • |
| | 上传软件包到软件发布库 上传软件包到软件发布库 上传软件包到软件发布库 | |
| | ● 添加步骤 | |

步骤6选择"添加步骤",单击"文件上传",选择"制作镜像并推送到SWR仓库",单击 "添加"。

图 10-56 添加步骤

| ← Cae-Dackend-SWF 基本信息 构態が確 仲間公論 例行け効 | |
|---|--|
| 50657 50658 BRANCESS BRANCESS | 添加学校 所有多常 原有时候 <u>文件上考</u> 市路用 Q (1000.110021000 |
| L REPORT FRANK | Mingarhaciównów Rozwowiewinawinaszowieże, zeskinaw |
| Maver Mavenlijk OfBoache MavenlijLavellij. | ■ 上代软件包括软件发布件 上代软件包括软件送示率 查看描述自由者 |
| S XXXX | LITZ/HOOS |

步骤7 进入"参数设置"页面,参考<mark>图10-57</mark>配置构建参数和默认值,并开启运行时设置。 "org_name"为上传到SWR镜像仓库的组织,"image_name"为上传到SWR镜像仓 库的镜像名称,"image_tag"为镜像标签或镜像版本。

图 10-57 配置自定义参数

| ← cae-backend-swr 基本信息 构建步骤 参数设置 計 | 行计划 | | | |
|------------------------------------|-----|-------------|--------|---------|
| 自定义参数 系统预定义参数 Q 请输入关键字 | | | | |
| 名称 | 类型 | 默认值 | 私憲參数 💿 | 运行时设置 🛞 |
| codeBranch | 字符串 | master | | |
| org_name | 字符串 | for-test | | |
| image_name | 学符串 | cae-backend | | |
| image_tag | 字符串 | v1.0 | | |
| + 新建参数 | | | | |

步骤8 进入"构建步骤"页面,配置"制作镜像并推送到SWR仓库"步骤,配置组织使用参数"\${org_name}",镜像名字使用参数"\${image_name}",镜像标签使用参数"\${image_tag}",工作目录使用默认值,Dockerfile路径按照代码仓中的路径来,示例中的Dockerfile在代码仓根目录。

| ← cae-backend-swr 基本信息 构建步 | · 参数设置 执行计划 | |
|----------------------------|---|---|
| | <u>四用4代</u> 代码4化 400日 年4月11日 400日 年4月11日 10日日 10日日日 10日 | 制作模像并推送到SWR仓库 通过Docardine和中制度ISWR仓库。查察保存和者 |
| | | - 参加型示点作 Monite 分析 * 工程に本 - 工程に本 - Sale 本 - Sale a - Sale a<!--</th--> |
| | Maven時間 使用Apache Maven院記Java時日。 | * 報告のホ © 単分式管理機構整整 単分式機構也成素がRR |
| | ····································· | * 認知市中 総和市中 |
| | Receive | * matisk @ |

图 10-58 配置"制作镜像并推送到 SWR 仓库"步骤

步骤9 单击右上角"新建并执行",确认运行时参数无误后,单击"确定"。

图 10-59 执行构建任务

| | 取消 | 新建 | 新建并执行 |
|------------------------|-----|----|-------|
| | | | |
| 执行 | × | | |
| 运行时参数 | - 1 | | |
| 名称 | | | |
| org_name for-test | | | |
| image_name cae-backend | | | |
| image_tag v1.0 | | | |
| 确定 取消 | | | |

步骤10 在"构建历史"页面选择构建任务查看构建日志,如<mark>图10-60</mark>所示,已构建成功并制作 镜像推送到SWR仓库。

图 10-60 查看日志

| ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ | 首页 / cae-teset / 编译构建 / 构 | 9建历史 / cae-l | backend-swr |
|--|------------------------------------|--------------|---|
| 伊護日志 构建参数 | ← ♥ #20231028.2 代码源: û cae-backend | 운 master 기 | 触发关源: ①) 手工設发 构建时间与时长: 曽 2023/10/28 18:05:00 GMT+08:00 ☉ 1m32s 关系: ⑤ 0 构建包下载 |
| ○ 代码检出 75 内健田志 ○ Maven均達 165 167 167 168 169 167 169 <l< th=""><th>步骤日志 构建参数</th><th></th><th></th></l<> | 步骤日志 构建参数 | | |
| 20 Mavenf3注 165 167 [2023/10/28] 18:06:30.426 (W1+08:09) [Delted::sha25-btf31040402/32ert80c7/9990d80.100Hb75408aa930be0b540561243442/322 203 制作镜像开描送到SWR 385 1677 [2023/10/28 18:06:30.426 (W1+08:09] Delted::sha25-660e04F2111052b181da4t39c425640495d1d46602.09341e015hc29545F7b0Ff 2023/10/28 18:06:30.665 (W1+08:09] Dilted::sha25-660e04F2111052b181da4t39c425640495d1d46602.09341e015hc29545F7b0Ff 2023/10/28 18:06:30.665 (W1+08:09] [IMF0] [M#t@g开推送到SMR仓催:external_docker_builder]: uplaad docker info success 2023/10/28 18:06:30.665 (W1+08:09] [IMF0] [M#t@g开推送到SMR仓催:external_docker_builder]: gbr@ht72i1052b181da4149c12640495d1d46602.09341e015hc29545F7b0Ff 2023/10/28 18:06:30.665 (W1+08:09] [IMF0] [M#t@g开推送到SMR仓催:external_docker_builder]: gbr@ht72in06 2023/10/28 18:06:30.667 (W1+08:09] arc (CWWT_Doc281921aa693db2258e14bae327ccf2e7818e718 2023/10/28 18:06:30.677 (W1+08:09] aar: (CWWT_Doc281921aa693db2258e14bae327ccf2e7818e718 2023/10/28 18:06:30.677 (W1+08:09] aar: (CWWT_Doc281921aa693db2258e14bae327ccf2e7818e718 2023/10/28 18:06:30.677 (W1+08:09] aar: (CWWT_Doc281921aa693db2258e14bae327ccf2e7818e718 2023/10/28 18:06:30.670 (W1+08:09] aar: (CWWT_Doc281921aa693db2258e14bae327ccf2e7818e718 2023/10/28 18:06:30.702 (W1+08:09] aar: (CWWT_Doc281921aa693db2258e14bae327ccf2e7818e718 | ✓ 代码检出 | 7s | 构建日志 |
| | 🥑 Maven构建 | 16s | 16/5 _{282/316/2,48} 16:06:30.426 6H1+08:061 Ueleted: sha2/50:16131664d642/9eeff6262200300100000000000000000000000000000 |
| 1633 [2623/10/28] 18:36:32.02 001-102_5001_120_500_120_500_500_500_500_500_500_500_500_500_5 | ⊘ 制作讓像并推送到SWR | . 385 | 107 [2427]/0/28 18:06:30.66.26 GHT.48:09 [INFO] [Bit @ght #Ed39MxCfg = setternal_docker_builder]: uploaded ocker info success. 1678 [2023]/0/28 18:06:30.66.57 GHT.48:09 [INFO] [Bit @ght #Ed39MxCfg = setternal_docker_builder]: upload ocker_brist 1679 [2023]/0/28 18:06:30.66.57 GHT.48:09 [INFO] [Bit @ght #Ed39MxCfg = setternal_docker_builder]: upload ocker_brist 1680 [2023]/0/28 18:06:30.677 GHT.48:09 as: COWUTI_Doc391221aa693db225414baa327ccf2e7810e718 1681 [2023]/10/28 18:06:30.702 GHT.48:09 as: COWUTI_Doc391271aa693db2258e14baa327ccf2e7810e718 1682 [2023]/10/28 18:06:30.702 GHT.48:09 as: COWUTI_Doc391271aa693db2258e14baa327ccf2e7810e718 1684 [2023]/10/28 18:06:32.131 GHT.48:09 as: COWUTI_Doc391271aa693db2258e14baa327ccf2e7810e718 1685 [2023]/10/28 18:06:32.131 GHT.48:09 as: COWUTI_Doc391271aa693db2258e14baa327ccf2e7810e718 1686 [2023]/10/28 18:06:32.131 GHT.48:09 as: COWUTI_Doc391271aa693db2258e14baa327ccf2e7810e718 1687 [2023]/10/28 18:06:32.155 GHT.48:09 as: COWUTI_Doc391271aa693db2258e14baa327ccf2e7810e718 168 |

步骤11 登录SWR控制台,在"我的镜像"中查看镜像已上传成功。

图 10-61 查看镜像

| 容器镜像服务 | 我的镜像 / cae-backend | |
|--|---|---|
| 参店 我的機像 機像资源 ▼ 相切管理 体验馆 企业版 (2) | · 镜像会称 cae-backend 类型 系有 反本数 1 已用空间 228.0 MB | |
| | 寶像版本 描述 PullPush指南 权限管理 共享 触发器 「検像同步 批量删除 」 | 镜像老化 镜像同步记录 大小 228.0 MB |

----结束

创建流水线构建后升级 CAE 组件

创建codearts流水线:

- 步骤1 返回Codearts控制台首页。
- **步骤2** 在菜单栏选择"服务 > 流水线",在"流水线"页面单击"新建流水线"。

| 图 10-62 | 选择服务 |
|---------|------|
|---------|------|

| ション 华为云 | 控制台 | | ■服务▼ | | | | |
|------------|--------|---|---------|-------------------|-------------|-----------|-----------|
| ae-teset | • | 首页 / cae-teset / 流水线 | | | Г | | 1 |
| - 1/ | | 流水线 策略 | | $\langle \rangle$ | | 1 | |
| _11 | | 금 ▲ 全部分组 + 新建流水线 全部流水线 ▼ Q 清输入器 | 架构建模 | 代码托管 | 代码检查 | 流水线 | 编译构建 |
| 构建模 | | 名称 | | | | | |
| 码 | > | #26 car | | 3 | 6 | 4 | 3 |
| 续交付 | \sim | Cae-demo P maste | 测试计划 | 部署 | 制品仓库 | Wiki | 文档 |
| 荒水线 | | | | | | | |
| 发布管理 | | | گ | - | | 512 | |
| 肩译构建 | | | 扩展插件 | IDE Online | IDE | DevStar 🤊 | APIOps >> |
| 部署 | | | | | | | |
| 运维 | | | 0 | 0 | Flow | | |
| 品仓库 | > | | 漏洞管理服务》 | 开源治理服务 🤊 | Astro 工作流 > | | |
| 0-2 | | | | | | | |

图 10-63 新建流水线

| 🤌 流水线 | | |
|----------------------------|--------|-----|
| 流水线列表 模板列表 | | |
| + 新建流水线 全部流水线 • Q 请输入流水线名称 | | |
| 名称 👅 | 最近一次执行 | 工作流 |
| | | |

步骤3 在"新建流水线"界面,输入"名称"并选择"流水线源"、"代码仓"和"默认分支",单击"下一步"。

| 横板 | 基本信息 | | | | | | |
|--------|--|----------|-------------------|--------|--------|--|---------|
| 19612X | *所屋项目 | | | | | | |
| | | | | | * | | |
| | *名称 | | | | | | |
| | swr_pipeline | | | | | | |
| | *流水线源 | | | | | | |
| | CA Proposition Providence Provide | GitHub | G 码云Git | Gerrit | GitLab | | |
| | ● 通用Git | GitCode | ● 通用Git | 暂不选择 | | | |
| | *代码仓 | | | | 刷新 | | |
| | cae-backend | | | | | | |
| | *默认分支 💿 | | | | | | |
| | master | | | | • | | |
| | Repo https授权 | 0 | | | 新建授权 | | |
| | 请选择 | | | | • | | |
| | 别名 👩 | | | | | | |
| | 别名仅支持输入 | 大小写英文字母、 | 数字、12,最多 | 128个字符 | | | |
| | 描述 | | | | | | |
| | 请填写流水线描 | 莇述 | | | | | |
| | | | | | | | |
| | | | | | | | 0 / 200 |
| | | | | | | | |

图 10-64 配置流水线基本信息

步骤4模板选择"无>空模板",单击"确定"。

| 基本信息 | 模板选择 | | 自定义模板 | 18 新建模板 | ≒ 收起 |
|-----------------------------------|---------|----|--|---------|------|
| | Q 搜索模板 | | template-20211026153921 | | |
| | 86 全部 | 20 | 开始 构建 结束 | | |
| | 를 Java | 8 | ConviceChange Marian (#HT 17799) convi | | |
| | Node.js | 5 | Servicestage-iwaven-应创品者-copy | | |
| | Python | 2 | 开始 代码检查 构建 部署 结束 | | |
| | ∞ Go | 1) | 系统模板 | | |
| | ● PHP | 1 | 空模板 | | • |
| | ↔ 其他 | 1 | 开始 结束 | | |
| | 回无 | 2 | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

步骤5 进入"任务编排"页面后,单击"新建阶段"。单击 2 将"阶段_1"改名为"构建","阶段_2"改名为"部署"。

图 10-66 新建流水线阶段

图 10-65 选择模板

| ← : | swr_pipeline 🖻 🛛 基本信 | :思 任务 | 编排 | 参数设置 | 执行计划 | 权限管理 | 通知 | 订阅 | | | | | | | | | | | | | |
|-------|----------------------|--|---------|----------|--------|------|--------|----------------|---|--------------|-----------|-----|-----|---|---------|-------|---|--|--|--|--|
| :25-1 | k 线 酒 | | 約建 | • | | | | 立 [[1]] | | | | | | | | | | | | | |
| 1 个 | 代码源 | Û | 1个 | 任务 | 0 | | Į II I | 0 个任务 | ş | | 0 | 0 (| נ | | | | | | | | |
| | | | | | | | | | | | | | | - | | | | | | | |
| | | •————————————————————————————————————— | <u></u> | ◎ 新建任 | 8 | | | | | + 新建在 | F\$ | | -0- | + | \$153£8 | l∩ £0 | | | | | |
| | 🐼 Repo | Ŭ, | | C WINELL | + | | | | | | | | Ŭ | | | | _ | | | | |
| | | | | | | | | | | | | | | | | | - | | | | |
| | 8º master | | | | | | | | | ピ 准田祭 | 条件 | | | | | | | | | | |
| | 别名: | | | н | + 并行任务 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | |
| | | | | | A | | | | | | | | | | | | | | | | |
| | | | | Ľ, | 5 准田孫件 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | |

步骤6 新建构建任务。

- 在"构建"阶段,单击"新建任务"。
 - a. 在"新建任务"中,单击"构建",选择"Build构建"任务,单击"添加"。

图 10-67 新建构建任务

| $ \leftarrow \ swr_pipeline \ \bowtie$ | 基本信息 任务编排 | 参数设置 执行计划 | 权限管理 通知订阅 | | | | |
|---|-----------|-------------|-----------------------|---------------|-----|--------|--|
| 流水线源 1 个代码源 | ŵ !! | 均建 | 口 前 # 部署 0 个任务 | R. D | 0 0 | | |
| 🐼 Repo | •—•• | ⑦ 新建任务 + | - • - 0 | + 新建任务 | ⊕- | + 新建阶段 | |
| 습 cae-backend 양 master 别名: | | + 并行任务 | | じ 准出条件 | | | |
| | | じ 准出条件 | | | | | |

图 10-68 添加 "Build 构建"任务

| 新建任务 🧷 | | i × |
|----------|--|--------|
| 请从右侧选择插件 | Q 请输入关键字 | 3 更多插件 |
| | 全部 内建 代码检查 部署 憲武 通用 Build构建 ③ 任务 调用CodeArts Build能力进行构建、编译构建服务(CodeArts Build)为开发者提… | 添加 |

b. 选择前一步创建的"编译构建任务"和"仓库",并将"org_name"、 "image_name"和"image_tag"都设置为流水线参数,单击"确定"。

图 10-69 image_name

| Build构建 🧷 | | 1 | × |
|-------------------|--|---------------------|----------------|
| 任务配置 Build构建 : | Build构建 调用CodeArts Build能力进行构建。编译构建服务(CodeArts Build)为开发者提供配置简单 名称 Build构建 | ① 温専 | 提示 F |
| | 请选择需要调用的任务 ◎ 技不到合适的任务?点击创握 cae-backend-swr 仓库 | ≧ <i>F</i> | 创新 ▼ |
| | Call-backend 参数名称: image_name 设置为流水线参数,支持任务中引用该参数 设置为流水线参数 | | • |
| | image_tag v1.0 | | |

图 10-70 image_tag

| Build构建 🧷 | Ω. | × |
|-------------------|--|---------|
| 任务配置 Build构建 : | Build构建 ① 温 调用CodeArts Build能力进行构建。编译构建服务(CodeArts Build)为开发者提供配置简单 慶 名称 | 器 研 |
| | buildrag建 • 请选择需要调用的任务 ● 找不到合适的任务?点击创建 ↓ cae-backend-swr • 仓库 cae-backend • org_ name | 副新 ▼ |
| | | |

c. 进入"参数设置"页面,添加自定义参数,如图所示,取值见<mark>参数说明</mark>。 "org_name":上传到SWR镜像仓库的组织,如for-test。

"image_name": 上传到SWR镜像仓库的镜像名称,如cae-backend。

"image_tag":镜像标签或镜像版本,如v1.0。

"swr_address": SWR镜像仓库地址,格式: swr.区域项目名 称.myhuaweicloud.com。

图 10-71 添加自定义参数

| swr_pipeline 🖯 基本信息 🕴 | 1844 9828 467113 (| RINNER MANTH | | | | | | 8 |
|-----------------------|---------------------------|--------------|---------------------------------|--------|---------------|------|-------|---|
| | 自定文学数 系统预定文学校 | | | | | | ①如何使用 | |
| | 88 | 東京 | \$F5A.00 | 私来争数 🛞 | រត់កាមខេត្ត 🔘 | 参数装送 | 101 | |
| | org_name | 字符串 | for-test | | • | | ÷ | |
| | image_name | 字符曲 | cae-backend | | | | ÷ | |
| | image_tag | 字符曲 | v1.0 | | | | ۲ | |
| | swr_address | 字符曲 | swr.cn-east-3.myhuaweicloud.com | | | | ۲ | |
| | + 1830 | | | | | | | |

- 在"部署"阶段,单击"新建任务"。
 - a. 在搜索框中输入"CAE",并单击 🔍 。选择"CAE发布"插件,单击"添加"。

图 10-72 添加部署插件

| 基本负息 任劳动的 参数设置 执行计划 彩质管理 通知订阅 | 新建任务 🖉 |
|---|--|
| B FNE / O B SHE / O B SHE / O B SHE | 第565前日前时 〇、CAE 〇 金星がられ 全部 時後 代記校告 容等 第4 週月 |
| + RIGER | CAE提布 • CAE (Cloud Application Engine完成研究等)是一个面积应用的Serveriena我是一 |

- b. 配置插件参数完成后,单击"确定"。
 - 区域选择:待部署的区域,与当前CodeArts在同一个Region。

- 环境:选择组件所属环境。
- 应用:选择组件所属应用。
- 组件:选择待升级的组件(需要选择在CAE中使用镜像部署的组件)。
- 组件来源:选择"镜像"。
- 镜像地址:取值见参数说明。

图 10-73 配置插件参数

| CAE发布 ⊘ | |
|--|--|
| CAE发布 ···································· | 三 ③ ☆ × ご 盗 業 CAE发布 ◆ V1.0.0 → ① 温馨提示 CAE (Cloud Application Engine云应用引擎) 是一个面向应用的Serverless托管服务,提供展开 *名称 CAE发布 |
| | *区域选择 *环境 *应用 |
| | ◆ ◆ ◆ ◆ ◆ ◆ ◆ |
| | ● 損像 ○ 软件包 链像地址 \${swr_address}/\${org_name}/\${image_name}:\${image_tag} |

c. 单击"保存并执行",进入执行配置页面,确认执行配置信息无误后,单击 "执行"。

| | 图 10-74 确认执行配置 执行配置 | | | × |
|-----------------|--|-------------|-------------------|---|
| | 流水线源 | | | |
| | 仓库 | 分支/标签 | | |
| | 🐼 cae-backend | 分支 🔻 | master | • |
| | ▼ 运行参数配置 | | | |
| | 名称 | 值 | | |
| | org_name | for-test | | |
| | image_name | cae-backend | I | |
| | image_tag | v1.0 | | |
| | swr_address | swr | myhuaweicloud.com | |
| | ▼ 执行阶段配置 ● ♥ 所有阶段 ● ♥ 构建 ● ♥ 合建 ● ♥ 部署 ● ♥ 部署 | | | |
| 步骤7 查看流水 | 线执行结果。 | | | |

图 10-75 查看流水线执行结果

流水线详情 执行历史 流水线源 构建 部署 0 0 0 1个代码源 1/1个任务完成 | 3m12s 1/1个任务完成 | 1m10s 逊 CloudBuild 3m12s -3 ✓ CAE发布 1m10s -🐼 Repo cae-backend ₽[®] master 别名:--

步骤8 登录CAE控制台,在组件列表查看组件状态,"最近一次变更状态/时间"列显示状态为"升级成功",则表示Codearts流水线执行升级成功。

----结束

参数说明

表 10-3 参数说明

| 参数名称 | 是否必须 | 参数类型 | 描述 |
|-----------------|------|--------|---|
| org_name | 是 | String | 镜像上传到SWR的组织名称,可以登录 <mark>SWR控制台</mark> 查看。 |
| image_na me | 是 | String | 上传到SWR的镜像名称。 |
| image_ta g | 是 | String | 上传到SWR的镜像版本。 |
| swr_addre ss | 是 | String | 值的格式:swr.区域项目名 称.myhuaweicloud.com,区域项目名称就 是region,参考 <mark>参数值获取</mark> 。 |
| 镜像地址 | 是 | String | 上传到SWR的镜像地址,值填写: \$ {swr_address}/\${org_name}/\$ {image_name}:\${image_tag}。 |

11 通过配置 PromQL 实现自定义弹性伸缩

假设有一个名为my_component的组件,组件所处环境为my_environment,所处应用为my_application。

查询指标

使用以下PromQL语句,查询最新一条指标数据:

http_requests_total{environment_name="my_environment",application_name="my_application",component_ name="my_component"}

若查询最近5分钟的所有指标数据:

http_requests_total{environment_name="my_environment",application_name="my_application",component_ name="my_component"}[5m]

处理查询到的指标

查询指标中查询的指标数据通常有多条,例如组件有多个实例,则指标数据也有多条,或查询了一段时间的指标数据,该段时间内采集多条数据。

伸缩策略中的PromQL必须返回单个值,因此需要处理查询得到的指标数据,以得到单个值,示例如下:

查询最新一条指标,并求平均值,得到所有实例的http请求总数的平均值:

avg(http_requests_total{environment_name="my_environment",application_name="my_application",compon ent_name="my_component"})

查询最近5分钟的所有指标数据,获取变化值(即增长值),并求平均,得到5分钟内 平均每个实例的http请求增长数:

avg(delta(http_requests_total{environment_name="my_environment",application_name="my_application",co mponent_name="my_component"}[5m]))

更多请参考PromQL官方文档和PromQL官方示例。

12 配置 CAE 对接 DEW,帮助应用从 DEW 获取加密凭据

12.1 概述

应用场景

CAE提供对接DEW帮助应用从DEW获取凭据的功能,简化应用与DEW对接的复杂度, 并提升安全性。

当您的代码里需要注入访问密钥、SSH Key等凭据作为环境变量时,可以通过配置 DEW凭据来保证密钥的安全性。

如何创建及管理凭据,请参考<mark>凭据管理</mark>。

文档版本 01 (2025-01-22)

方案架构



图 12-1 凭据注入环境变量的流程图

约束限制

此功能需要您授权KMS CMKFullAccess以及CSMS ReadOnlyAccess到cae_trust委托中,具体操作请参考给用户组授权。

在启用此功能前需要使用管理员用户进行DEW授权。

12.2 配置凭据,通过环境变量注入

本实践通过添加对应DEW凭据,以环境变量方式注入到组件内,帮您实现数据保护。

创建 DEW 凭据

- 步骤1 登录DEW控制台。
- 步骤2 在左侧导航树中,选择"凭据管理",进入"凭据管理"页面。
- 步骤3 单击"创建凭据",在弹框中参考表12-1填写参数。

表 12-1 凭据配置参数说明

| 参数名称 | 参数说明 |
|------|-----------------------|
| 凭据类型 | 选择通用凭据。 |
| 凭据名称 | 填写创建凭据的名称。本实例凭证名称为db。 |

| 参数名称 | 参数说明 |
|-------|------------------------------------|
| 企业项目 | 创建凭据时,凭据绑定企业项目ID。 本实例选择default。 |
| 设置凭据值 | 选择明文,输入123456。 |
| 描述信息 | 本实践无须填写。 |
| KMS加密 | 可选择默认密钥"csms/default" |
| 关联事件 | 本实践暂不关联。 |

- **步骤4**单击"下一步",通用凭据不支持设置轮转周期,再次单击"下一步",确认创建的 凭据信息。
- 步骤5 单击"确定",凭据创建完成。

用户可在凭据列表查看已完成创建的凭据,凭据默认状态为"启用"。

----结束

添加凭据配置

- 步骤1 登录CAE控制台。
- 步骤2 在左侧导航栏中选择"系统设置"。

如未授权KMS CMKFullAccess和CSMS ReadonlyAccess,请使用管理员账号进行委托 授权。

图 12-2 授权委托

授权给CAE

 \times

您确定要同意授以下权限给CAE吗?同意授权后,CAE将在统一身份认证服务为您创建名为 cae_trust 的委托,为保证服务正常使用,在使用CAE服务期间,请不要删除或者修改 cae_trust 委托。收起 ▲

| 权限名称 | 描述 |
|---------------------|---------------------|
| CSMS ReadOnlyAccess | CAE凭据配置需要获取您在凭据管理服 |
| KMS CMKFullAccess | CAE配置类型为凭据导入的环境变量时… |
| | |



步骤3 单击"凭据配置"模块中的"编辑",进入"凭据配置"页面。

步骤4 单击"创建凭据配置",在弹框中选择创建DEW凭据中已创建的凭据以及所需版本。

×

图 12-3 新增凭据

| 新谓先据 | 北直 | |
|------|----|---------------|
| 凭据名称 | db | ▼ C 前往DEW创建凭据 |
| 凭据版本 | v9 | • |
| | 确定 | 取消 |

步骤5 单击"确定",完成凭据配置。

----结束

配置环境变量

- 步骤1 在左侧导航栏中选择"组件配置"。
- 步骤2 在"组件配置"界面,选择需要配置的组件。
- 步骤3 单击"环境变量"模块中的"编辑",进入"环境变量配置"页面。
- 步骤4 单击"新增环境变量",考表12-2进行参数配置。

表 12-2环境变量配置

| 参数 | 参数说明 |
|---------|--|
| 类型 | 选择"凭据导入"。 |
| 变量名称 | 环境变量的名称,例如test1。 名称必须唯一,不可重复。 |
| 变量/变量引用 | 在下拉框中选择 <mark>添加凭据配置</mark> 中已创建的凭据配置。 |

图 12-4 配置环境变量

环境变量配置

| 🛕 请您在配置环境变量时慎用敏感信 | 息或者进行敏感信息加密,以免造成信息泄露 | 晷。例如:用户个人隐私、数据库密码等。 | |
|-------------------|----------------------|----------------------------|-------|
| | | | |
| ⊕ 新增环境变量 导入 | 批量删除 | | |
| | | | |
| 类型 | 变量名称 | 变量/变量引用 | 操作 |
| 凭据导入 ▼ | test1 | db 💌 | 保存 取消 |

步骤5 在"操作"列单击"保存",并在"环境变量配置"页面单击"确定",完成环境变量配置添加。

步骤6单击页面上方"生效配置"。 在右侧弹框中确认配置信息,并单击"确定",使配置生效。 ----**结束**

查看配置生效

- 步骤1 在左侧导航栏中选择"实例列表"。
- 步骤2 在"实例列表"页面上方的下拉框中选择环境、应用及待操作的组件。
- 步骤3 选择待操作实例,在"操作"列单击"远程登录"。
- 步骤4 查看具体环境变量,与DEW凭据中设置的凭据一致。

图 12-5 DEW 服务中的凭据值

查看凭据值

图 12-6 远程登录里的环境变量

| user@si | <pre>machine:~\$ kubectl -n</pre> | -env20231229 exec -it demo-frontend | 6956968d8-mx9wh -c demo-frontend-1704597891 /bin/sh |
|------------------|-----------------------------------|-------------------------------------|---|
| sh-5.0# env gr | ep test1 | | |
| test1=123456 | | | |
| sh-5.0# | | | |
| | | | |
| | | | |
| | | | |

----结束

12.3 屏蔽子账户读取密钥的权限,实现密钥保护

当您使用多账号管理CAE环境时,可以根据企业的业务组织,在您的华为云云服务账 号中,给企业中不同职能部门的员工创建子账号,并根据职能设置不同的访问权限, 以达到用户之间的权限隔离。

例如:分别为开发及测试创建开发账号与测试账号,测试人员无需感知敏感信息,为 保护敏感信息,您可以屏蔽测试账号获取敏感信息的权限。

 \times

本最佳实践指导您通过屏蔽子账号DEW服务的所有权限以及CAE远程登录的权限,来 控制子用户读取密钥。

创建自定义策略

- 步骤1 登录"统一认证服务"控制台。
- 步骤2 在左侧导航栏中选择"权限管理 > 权限"。

步骤3 单击"创建自定义策略",进入"创建自定义策略"页面。

图 12-7 进入统一认证服务页面

| 🗏 🎂 华为云 | 12862 | | | | 0.85875899.193867.298.211.491 846 837 887 44 IN IN D Q 🖓 🕀 944 1 |
|-------------------------|----------|---------------------------|--------------|--------------|--|
| 统一身份认证服务 | EXAR () | | | | © se eliteixae |
| л.» Л."Ц 621112 • | 200 2171 | - MNEED - | MADEX . | Q KHISH. Hid | |
| 165.187B | - 8# | | A 2 | Sid | 愛介 |
| 631 | | | 自定义策略 | | and also |
| ALCONG DAY | | | R02.0308 | | 846 (20) |
| dir5 | | | 自己义策略 | | 56 59 |
| 自己经济常 | | | 08.5%M | | 85 20 |
| 2992B | | | 自定公開等 | | #G 100 |
| | | | 0.2.0.04 | | 68, 89 |
| | | | 05.594 | | 86 89 |
| | | | 和技术制度 | | 80 |
| | | | R25388 | | 46 59 |
| | • | | 自己文策略 | | 66 89 |
| | 11 × 850 | 2,505 (1 2 3 4 5 6 ~ 251 | > | | |

步骤4 配置自定义策略。

- 策略名称:填写cae-subuser。
- 策略内容:选择"拒绝"。
- 服务:选择"云应用引擎"。
- 操作:勾选 "cae:application:createConsole"权限。

图 12-8 创建自定义策略

| 权限 / 创建自定义策略 | | | | | |
|--------------|-----------------------------------|----------------------------------|----------------------------------|---------------------------------------|-----|
| 自定义策略是对系 | 统策略的扩展和补充,您可以创建自定义策略以实现定 | 制的权限管理。 ② | | | |
| | | | | | |
| ★ 策略名称 | cae-subuser | | | | |
| 策略配置方式 | 可視化視躍 JSON视题 | | | | |
| * 策略内容 | ヘ ③ 拒绝 | C 云应用引擎 | C 1项操作 | ③ 请求条件 (可选) | Ū Ū |
| | 选择所有操作 请输入关键字 | | | Q | |
| | ◇ □ 只读 共2项操作 | | | | |
| | ヘ ■ 写 共8项操作 已选择1项 | | | | |
| | Cae:environment:delete 删除环境 | Cae:application:modify 更新应用信息 | Cae:environment:modify 更新环境信息 | Cae:environment:update 更新环境信息 | |
| | Cae:environment:create 创建环境 | Cae:application:create 创建应用 | Cae:application:delete 删除应用 | Cae:application:createConsole 远程登录 | |
| | ◇ □ 列表 共2项操作 | | | | |
| | ④ 从已有策略复制 ④ 添加权限 | | | | |
| 策略描述 | 请输入策略描述 (可选) | | | | |
| | | | | | |
| | | 0/2 | -∑ 56 | | |
| 作用范围 | 项目级服务 | | | | |
| | 発定 取消 | | | | |

步骤5 单击"确定",完成自定义策略创建。

----结束

创建用户组并授权

步骤1 在左侧导航栏中选择"用户组",单击"创建用户组"。



| 统一身份认证服务 | 用户题 ① | | | 91877-10 |
|-----------------|-------------------------------|----------|-----------|-------------------------------|
| m+ | | | | |
| m-se | 2010 2012/07/08/2010/17/2010L | | | |
| 638#38 v | Q 诸侯人现外的名法日期来 | | | |
| 100 | 1 #ABA# : | ROBE SEC | essesia ÷ | 間作 |
| #6% | | | | 接收 網羅 用户接管理 動脉 |
| 非公認不問 | | | | 907 908 A-0222 809 |
| 910A | | | | 2012 2018 R/201178 2010 |
| | | | | · 1972 · 668 · 月中世世世 · 世际 |
| | | | | 1992 594 70-9252 899. |
| | | | | 10次(66篇)从小台的进一步脉 |
| | | | | 1957 编辑 用户指数理 数除 |
| | | | | 1952 1968 12-1012 1810 |
| | | | | 10次(編集)用户加加进一般标 |
| | • | | | 887 968 RACESS 889 |
| | 10 • 8500:14 < 1 2 > | | | |

步骤2 输入"用户组名称",例如cae-test,单击"确定",完成用户组创建。

步骤3 在用户组列表中单击创建完成的用户组名称"cae-test",进入用户组概览页面。

步骤4 在"授权记录"页签中单击"授权",选择创建自定义策略中创建的自定义策略。

图 12-10 授权自定义策略

| 授权 | |
|--|----------------|
| 选择策略 ② 设置最小授权范围 ③ 完成 | |
| 用户组"cae-test"将拥有所选策略 ⑦ | |
| 查看已选(1) 从其他区域项目复制权限 | 全部类型 ▼ 所有云服务 ▼ |
| 各称 | 类型 |
| □ ✓ policy987h16 - | 自定义策略 |
| Cae-subuser | 自定义策略 |
| □ ~ ^{vpc} | 自走义策略 |
| □ | 自定义策略 |

步骤5 单击"下一步",选择授权范围方案为"所有资源"。

步骤6 单击"确定",完成授权。

待授权完成后,单击"完成",返回用户组概览页面。

图 12-11 完成用户组授权

| 统一身份认证服务 | 用户编译 / coentext | | | | | | 809 |
|----------------------|---------------------------|-------------------------------|----------------|----------|------------|------------------------|--------|
| 84 848 67888 • | RPERE cartail & | 現の版D 化確が用 | | | | | |
| 25 26322 1202 | BECCOR PERMIT | ACC2 AVEL Avenue Avenue | | | | | |
| | 200 800 2012230743 | AND RECEIPTS. 2100 RECEIPTS | | | RMBE cashe | A DESCRIPTION Q MARTIN | 222868 |
| | 638 | 82813618 | 项目(作用ISM) | 1957.238 | 主体疑述 | 134天型 四十 | |
| | Case subacer | | 所有出现(也会未伸新爆攻目) | can-test | | R~6 80 | · |
| | | | | | | | |

步骤7 单击"用户管理",进入"用户组管理"页面。

步骤8 在用户列表中勾选需要控制权限的子用户,单击"确定"。

图 12-12 配置用户管理

| 用户组 / cae-test | | | | |
|------------------|----------------|-----------|------------|-----------|
| 用户组名称 cae-test 🖉 | 用户组管理 | | | × |
| 描述 🖉 | 用户组名称 cae-test | | | |
| | 可选用户 (共50条) | 请输入用户名称 Q | 已选用户 (共3条) | 请输入用户名称 Q |
| | ■ 用户名称 | 所属用户组 | 用户名称 | 操作 |
| 授权记录用户管理 | | 查看 | | × |
| 添加移除 | | 查看 | | × |
| 用户名称 ≜ | | 查看 | | × |
| | | 查看 | | |
| | | 2 查看 | | |
| | | 查看 | | |
| | | 查看 | | |
| | | 查看 | | |
| | | 查看 | | |
| | | 查看 | | |
| | 10 🔻 < 1 2 3 | 4 5 > | | |
| | | | | |
| | | 确定 | 取消 | |
| | | | | |

----结束

验证子账号权限

步骤1 使用步骤8的子用户账号登录CAE控制台。

步骤2 进行**添加凭据配置、配置环境变量**等操作,可正常使用。

步骤3 在左侧导航栏中选择"实例列表"。

步骤4 选择待操作实例,在"操作"列单击"远程登录",无法查看凭据具体内容。

图 12-13 当前子用户无远程登录的权限

| 💮 лапон | 🐼 34: • • • 🛞 64: | • | | ● CAL-01500002 用户用号使用子型、偏位型用号型用有相数 × 发展作使用 |
|--------------|-------------------|---------------------------|-------------------------------|---|
| ● 新成 編件利用 | • iB4: /v100 • | | | |
| 2010 | \$291247 c | 秋华 Δ | dutatil c | 時代 |
| 由中配量 | | 2015年 | 2024/01/11 11:36:03 GMT+00:00 | 5692R 899 |
| 经计算计 | | | | |
| 相性监视 | | | | |
| 经计目表 | | | | |
| 系统花園 | | | | |
| | | | | |

步骤5 单击左上角 —, 在服务列表单击"数据加密控制台 DEW",切换至DEW控制台。 **步骤6** 在左侧导航栏中选择"凭据管理 > 凭据列表",无法查看凭据具体内容。

图 12-14 当前子用户无 DEW 的委托权限

| 教派加密控制合 の | 発展管理 ② | | | | | (1) 20095767-32. Rendematication permission to perform action or | NGREED. You do not have \times smscills/Events on resource *. |
|--|--------------------|-------|--|----------|-------|---|---|
| 如何想道 如何可想道 | *258 ***** | | | | | | |
| · 元序包括 · · · · · · · · · · · · · · · · · · · | Q SPRENCE ERAAMPER | W25 0 | 40000 0 | esteni : | 00381 | 50 | C @ |
| 中日進均 中国10日 中国10日 中国10日 中国10日 大田市中国10日 王田市日 王田市日 王田市日 | 0 v #882 0 < 1 > | | 17200 Lewiger Strukter House (State | MERFAIR. | | | |
| 结 | 束 | | | | | | |

13 ASP.NET Core 应用部署到 CAE

CAE源码部署支持Docker运行时,您可以自行配置Dockerfile文件,在Dockerfile中安装构建环境,定义构建命令,以此支持更多编程语言的项目在CAE部署。

本文以ASP.NET Core应用为例。

前提条件

已创建环境和创建应用。

Fork 示例源码

使用您的账号登录GitHub,并Fork示例源码仓库到个人账号。

源码地址: https://github.com/Azure-Samples/dotnetcore-docs-hello-world。

Dockerfile 解析

示例仓库中的Dockerfile文件。

指定基础镜像为mcr.microsoft.com/dotnet/sdk:7.0,该基础镜像作为ASP.NET Core项目的构建环境 FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build # 指定工作目录 WORKDIR /source

从构建主机上复制源码到基础镜像,并指定工作目录和构建命令 COPY . ./dotnetcore-docs-hello-world/ WORKDIR /source/dotnetcore-docs-hello-world RUN dotnet publish -c release -o /app

指定新的基础镜像为mcr.microsoft.com/dotnet/aspnet:7.0,并将第二步生成的构建产物复制到新的基础镜像 中 FROM mcr.microsoft.com/dotnet/aspnet:7.0 WORKDIR /app COPY --from=build /app ./

设置环境变量PORT,并声明容器端口为80 ENV PORT 80 EXPOSE 80

指定容器启动命令 ENTRYPOINT ["dotnet", "dotnetcoresample.dll"]

设置 GitHub 仓库授权

设置GitHub仓库授权,使构建工程、应用组件等可以使用授权信息访问GitHub源码仓 库。

- 步骤1 登录CAE控制台。
- 步骤2 在左侧导航栏中选择"系统设置"。
- 步骤3 单击"源码仓库授权"模块中的"编辑",进入"已授权源码仓库"页面。

步骤4 单击"新增授权",参考表13-1选择需要的源码仓库,并配置参数。

表13-1 授权参数说明

| 参数 | 说明 |
|------|--|
| 授权名称 | 授权名称,创建之后不可更改 |
| 仓库类型 | 支持以下官方仓库类型: 1. 选择GitHub代码仓库。 2. 选择"授权方式"。 - OAuth:请单击"使用OAuth授权",根据页面提示完成授权创 建。 - 私人令牌:输入已获取到的私人令牌。 |

步骤5 单击"确认",完成GitHub代码仓库授权。

图 13-1 授权 GitHub 代码仓库

已授权源码仓库

| CodeArts | GitHub GitCode GitLab Gitee Bitbucket |
|------------|---------------------------------------|
| Github是一家; | 原代码托管网站,提供商业计划和免费帐户 |
| * 授权名称 | github-cr2hio |
| 授权方式 | OAuth 私人令牌 ⑦ |
| ★私人令牌 | ····· @ |
| | 确认 取消 |

----结束

云应用引擎 最佳实践

- 步骤1 在左侧导航栏选择"组件列表"。
- 步骤2 在页面上方,下拉选择已创建的应用和环境,单击"新增组件"。

步骤3 参考表13-2设置组件信息。

表13-2 组件基本信息

| 参数 | 说明 |
|------|---|
| 组件名称 | 新建组件的名称。本实践输入名称为"test-dotnet"。 |
| 版本号 | 组件的版本号。 本实践版本号为1.0.0。 |
| 实例规格 | 选择实例规格,例如:0.5core、1GiB。 |
| 实例数量 | 输入实例数为1。 |
| 代码源 | 选择"源码仓库 > GitHub"。选择 <mark>设置GitHub仓库授权</mark> 中创建好 的授权,选择示例代码仓。 |

图 13-2 配置基本信息

| 组件名称 | test-dotnet | | | | | | | | |
|------|-------------------|--------------|----------|--------|--------|-------|-------|---|-----------|
| 版本号 | 1.0.0 | | | | | | | | |
| 实例规格 | 0.5 Core | | • | 1 GiB | | • | | | |
| 实例数量 | - 1 + | | | | | | | | |
| 代码源 | 源码仓库 | 镜像 | | 软件包 | | | | | |
| | CodeArts | \mathbf{O} | GitHub | | GitLab | G | Gitee | 0 | Bitbucket |
| | | | <u> </u> | | | | | | |
| | Github 是一家 | 家源代码托给 | 管网站, | 提供商业计划 | 和免费帐户 | | | | |
| | 授权信息 | | | • | C 新建排 | 釵 或查看 | 授权列表 | | |
| | 用户名/组织 | | | • |] | | | | |
| | 仓库名称 | | | . • |] | | | | |
| | 分支 | master | | • |] | | | | |

步骤4 语言/运行时选择"Docker",Dockerfile地址填写./Dockerfile.linux。

图 13-3 配置 Dockerfile

| 语言/运行时 | Docker | | • | |
|----------------|--------------------|------|---------|---|
| 自定义构建 | 使用默认命令 | 或脚本 | 使用自定义命令 | ? |
| Dockerfile | 自定义 | 系统默认 | | |
| Dockerfile地址 ? | ./Dockerfile.linux | | | |

- 步骤5 单击"配置组件",跳转至"组件配置"页面。
- 步骤6 找到"访问方式",单击"编辑"。
- **步骤7** 在"从环境外部访问本组件"页面,选择"负载均衡配置",并单击"添加负载均衡 配置"。

参考图13-4配置负载均衡。

图 13-4 配置负载均衡

| 新增负载均衡配置 | | | | | | | |
|----------|---------|---|--|--|--|--|--|
| 负载均衡器 | 内置负载均衡器 | • | | | | | |

| 负载均衡器 | 内置负载均衡器 🔹 | | | |
|-------|---------------------|--------------------------|-------|----|
| 健康检查 | 不启用 启用 | | | |
| | 协议: TCP 检查周期(秒):5 超 | 时时间 (秒) : 10 最大重试次数: 3 🖉 | 2 | |
| 访问控制 | 允许所有IP访问 ▼ |] | | |
| 端口配置 | 协议 | 监听端口 | 访问端口 | 操作 |
| | TCP 💌 | 80 | 10000 | 删除 |
| | (+) 添加端口配置 | | | |

步骤8 单击"确定",完成负载均衡配置。

步骤9 单击组件配置页面上方的"配置并部署组件",在右侧弹框中单击"确定",待部署 执行完成后,配置生效。

----结束

查看应用

- 步骤1 在左侧导航栏,选择组件列表,返回组件列表页面。
- 步骤2 单击"test-dotnet"组件"访问地址"列的公网访问地址。

| 組件 | | | | | | |
|-----------------------|-------------------------------------|-------|--------------|--------------------------------------|---|-------------------------------|
| 名称版本 👙 | 代码源 🍞 | 秋恋 🍞 | 实例个数 (可用/全部) | 访问地址 | 最近一次变更状态时间 👙 | 创建时间 🖕 |
| test-dotnet v1.0.0 | 源码仓库 dotnetcore-docs-hello-world | 😏 运行中 | 1/1 | □ http://121.36.201.123:13689 (公网地址) | 部署成功 Q 2024/04/25 16:22:16 GMT+08:00 | 2024/04/25 16:19:41 GMT+08:00 |

步骤3 您还可以访问应用的静态页面。

dotnetcoresample Home Privacy

Hello World from .NET 7 Learn about building Web apps with ASP.NET Core.

Host operating system: Linux 4.18.0-147.5.1.6.h1071.eulerosv2r9.x86_64 #1 SMP Sat Jul 1 09:00:26 UTC 2023



14 通过企业路由打通网络

CAE提供CAE环境访问租户侧VPC的能力,本章最佳实践将指导您在关闭公网后,如何 通过企业路由打通网络。

步骤一: 创建企业路由并添加连接

创建企业路由器,并在企业路由中添加连接,本案例的vpc-wc为CAE环境所属VPC(网段为192.168.0.0/16),vpc-10为租户侧VPC(网段为10.0.0.0/16)。

- 步骤1 进入企业路由器控制台。
- **步骤2**单击"创建企业路由器",在创建企业路由器页面,按照如下参数示例进行设置,参数的具体含义请参考创建企业路由器。

| 耒 | 14-1 | 创建企业路由器配置参数 | |
|----|------|-------------|--|
| 1× | | 的建业业时间的自己多效 | |

| 参数 | 本案例示例 |
|----------|---------|
| 区域 | 华北-北京四 |
| 可用区 | 可用区3 |
| 企业项目 | default |
| 名称 | er-878b |
| ASN | 64513 |
| 默认路由表关联 | 开启 |
| 默认路由表传播 | 开启 |
| 自动接受共享连接 | 关闭 |
| 标签 | - |
| 描述 | - |

- 步骤3 设置完成后,单击"立即创建 > 提交"。
- **步骤4** 在企业路由器页面,单击刚才创建的企业路由器的名称"er-878b",选择"连接"页签。

步骤5在"连接"页签下,单击"添加连接",在添加连接页面,按照表14-2和表14-3参数 示例进行设置,参数的具体含义请参考在企业路由器中添加VPC连接。

表 14-2 添加连接至租户侧 VPC

| 参数 | 本案例示例 |
|---------|-----------------------------|
| 名称 | er-attach-374d |
| 连接类型 | 虚拟私有云(VPC) |
| 连接资源 | ● 虚拟私有云: vpc-10 |
| | • 子网: wtr-test(10.0.1.0/24) |
| 配置连接侧路由 | 开启 |
| 高级配置-描述 | - |
| 高级配置-标签 | - |

表 14-3 添加连接至 CAE 环境所属 VPC

| 参数 | 本案例示例 |
|---------|--|
| 名称 | er-attach-f855 |
| 连接类型 | 虚拟私有云(VPC) |
| 连接资源 | 虚拟私有云: vpc-wc 子网: wtr-test(10.0.1.0/24) |
| 配置连接侧路由 | 开启 |
| 高级配置-描述 | - |
| 高级配置-标签 | - |

步骤6 设置完成后,单击"立即创建"。

| < er-878b 基本信息 连接 路由表 非 | 導 流日志 标签 | | | | | | | 0 |
|--|---|--|--------|------------|-------------------|------|-------------|----|
| ● 当前将虚拟私有云置接接入企业指标 企业前由接支持接入多种网络支利 | 1番約建坦网封,存在部分使用限制,为了建 1应的运输,构建不可关型的祖同,了解50月 | 免疫的网络不通,商业看企业面也器吧网络 5在企业路由器中原始编统 (2 | 19 and | | | | | × |
| | | | | | | | | 00 |
| SERVID 0 | 87.05 O | 连接类型 谷 | 连接资源 0 | 所國版号/項目印 🖯 | XWEAR 0 | 顕述 🖯 | 10/11 | |
| er-attach-374d 89689321-1613-4650-966f-3d5de3 | ● 正常 | 虚拟私育云 (VPC) | vpc-10 | 当前符号 | defaultRouteTable | - | 修改 删除 | |
| er-attach-1855 196e2cab-8c9b-417b-8b81-d2996b | ⊘ 正常 | 遗拟私有云 (VPC) | урс-ию | 当前账号 | defaultRouteTable | - | 1972X 18579 | |

----结束

步骤二: 创建 NAT 网关

购买公网NAT网关,其中虚拟私有云为CAE环境所属VPC,在本案例中,CAE环境所属VPC为vpc-wc。

文档版本 01 (2025-01-22)

步骤1 进入公网NAT网关控制台。

步骤2 单击"购买公网NAT网关",在购买公网NAT网关页面,按照如下参数示例进行设置,参数的具体含义请参考购买公网NAT网关。

表14-4 公网 NAT 网关配置参数

| 参数 | 本案例示例 |
|---------|-----------------------|
| 区域 | 华北-北京四 |
| 计费模式 | 请您按照业务需要选择计费模式。 |
| 规格 | 请您按照业务需要选择公网NAT网关的规格。 |
| 名称 | nat-7a60 |
| 虚拟私有云 | vpс-wc |
| 子网 | cae-subnet |
| 企业项目 | default |
| 高级配置-描述 | - |
| 高级配置-标签 | - |
| 购买时长 | 请您按照业务需要选择购买时长。 |

步骤3 设置完成后,单击"下一步 > 去支付",支付后完成创建公网NAT网关。

| 公网NAT网关 ⑦ | | | | | | | | 🗋 使用指响 | 购买公网NAT网关 |
|--|-------|---------|------------|----------------|------|------|--------------------------------|-----------------------|-----------|
| 导出 > Q. 选择量性筛选,或输入关键字搜索 | | | | | | | | | 0 |
| Stand ⊕ | tta ⊖ | 虚拟私有云 ⇔ | 子問名称 令 | 子间网段 令 | 規格 ⇔ | 細述 令 | 计费模式 ⇔ | 操作 | |
| nat-7a60 2582ac2a-7f2f-47f3-8fae-affd06 | 3 运行中 | vpc-wc | cae-subnet | 192.168.0.0/24 | 小型 | | 按需计题 2024/08/10 15:13:06 创建 | 12 11 140,01 H | 改更多~ |

- **步骤4** 在公网NAT网关列表,找到刚才创建的网关"nat-7a60",单击右侧的"设置规则"。
- **步骤5** 在SNAT规则页签中,单击"添加SNAT规则",在添加SNAT规则页面,按照如下参数 示例进行设置,参数的具体含义请参考<mark>添加SNAT规则</mark>。

表14-5 添加 SNAT 规则配置参数

| 参数 | 本案例示例 |
|---------|-------------------------------|
| NAT网关名称 | nat-7a60 |
| 使用场景 | 虚拟私有云 |
| 网段 | cae-subnet (192.168.0.0/24) |
| 公网IP类型 | 100.95.154.131 |
| 监控 | - |

| 参数 | 本案例示例 |
|----|-------|
| 描述 | - |

步骤6 设置完成后,单击"确定"。

| < nat-7a60 | | | | | | | ## 52191648 |
|-------------------------------------|-------------------------|--------|------------------------------|----------------|-------|-------------------------------|--------------------|
| 基本信息 SNAT規則 DNAT規則 | 监控 标签 | | | | | | |
| 天田は同internet、清添加SNAT初期 | 7620103830550418034 | | | | | | |
| (添加SNAT規則) (导出 ~) | | | | | | | |
| 运择屬性夠造,或給入关键字搜索 | | | | | | | 0 |
| ID 0 | 402 O | 使用场景 ⊖ | MIG O | imte公网P ↔ | Hit O | annen 0 | 操作 |
| 47b6ae62-ac43-4265-b65a-ab6c | 运行中 | 虚拟私育云 | 192.168.0.0/24 cae-subnet | 100.95.154.131 | - | 2024/08/16 15:27:28 GMT+08:00 | 修改 删除 |

----结束

步骤三:租户侧创建 ECS 节点

租户侧创建的ECS节点,所属虚拟私有云为vpc-10,该节点弹性公网IP为 100.93.0.237,私有IP为10.0.0.80。

- **步骤1**进入。
- **步骤2**单击"购买弹性云服务器",在购买弹性云服务器页面,请您按照业务需要设置购买参数,参数的具体含义请参考购买ECS指南。

🛄 说明

在网络设置模块,设置"虚拟私有云"为租户侧VPC,在本案例中,租户侧VPC为vpc-10,该节 点弹性公网IP为100.93.0.237,私有IP为10.0.80。

步骤3 设置完成后,单击"立即购买"。

□ crs+2/21 回 (st-14/d0 bca0.2... 回) (crs+2/21 (crs+2

----结束

步骤四: 创建路由表和路由

在CAE所属VPC(vpc-wc)的路由表中添加路由,其中10.0.0.0/16为私网网段,100.93.0.0/16为公网网段。

- 步骤1 进入企业路由器控制台。
- 步骤2 单击刚才创建的企业路由器的名称 "er-878b" ,选择 "路由表" 页签。
- **步骤3**单击"创建路由表",按照如下参数示例进行设置,参数的具体含义请参考创建路由 表。

表 14-6 创建路由表配置参数

| 参数 | 本案例示例 |
|----|------------|
| 名称 | rtb-vpc-wc |
| 描述 | - |

| 参数 | 本案例示例 |
|----|-------|
| 标签 | - |

步骤4 设置完成后,单击"确定"。

步骤5 在路由表页面,选择刚才创建的路由表"rtb-vpc-wc",在"路由"页签下,单击 "创建路由",您需要按照如下两个参数示例表14-7和表14-8进行设置,参数的具体 含义请参考创建静态路由。

表 14-7 创建路由配置参数(下一跳为企业路由器)

| 参数 | 本案例示例 |
|------|------------|
| 目的地址 | 10.0.0/16 |
| 黑洞路由 | 不开启 |
| 连接类型 | 虚拟私有云(VPC) |
| 下一跳 | er-878b |
| 描述 | - |

表 14-8 创建路由配置参数(下一跳为 NAT 网关)

| 参数 | 本案例示例 |
|------|---------------|
| 目的地址 | 100.93.0.0/16 |
| 黑洞路由 | 不开启 |
| 连接类型 | 虚拟私有云(VPC) |
| 下一跳 | nat-7a60 |
| 描述 | - |

步骤6 设置完成后,单击"确定"。

| (invite-we | | | | | | | | |
|--------------------------------------|-------------------------|--------------------|--------------------------|--------|-------------------|------------------------------|--|--|
| 富息 关联子网 | | | | | | | | |
| | | | | | | | | |
| S₩ rtb-vpc-wc Z | | | 类型 | 默认語由表 | | | | |
| D fd855b11.#68-40ee-a302-41c70bc7db7 | 30 | | 虚拟私有云 | vpc-wc | | | | |
| 2022/05/26 10:19:15 GMT+08:00 | | | 描述 | ll. | | | | |
| | | | | | | | | |
| 路由 | | | | | | | | |
| 000 (#1/#tg) (#1/#tg) | But y | | | | | | | |
| | | | | | | | | |
| ○ 选择屋住闲选,或输入关键字搜索 | | | | | | | | |
| | 1000404.00 | | | | | | | |
| 日的地址 | IPRER | | 下一課 | | 東里 ⊖ | 描述 ⊖ | 操作 | |
| Local | PREED | 6 Local | Local | | 業理 ⊖ | 描述 ↔ 系统默认,表示VPC内实例互通 | 操作 | |
| Local | 11 ⁻¹ 85/123 | 6 Local | ▶→厩 Local | | 東型 ⊖ 系統 | 描述 ↔ 系统默认,表示VPC内实例互通 | 操作 | |
| Einfnitzh.t Local 10.00.0/16 | 11-78.92.0 | 6 Local 1 企业路由器 | t-⊒€ Local er-878b | | 素服 ⊕ 系統 自定义 | 描述 ⊕ 系统默认,表示VPC内实例互通 | 日日 「「「」」 「「」」 「」」 「」」 「」」 「」」 「」」 | |
| Exhibits Local 10000116 | 1998.23 | 6 Local 1 企业统由融 | k−æ | | 案目 ⊕ 系統 創定义 | 国語 ⊕ 系统就认、参示VPC内实例回通 一 | 操作 作改图称 作改图称 | |

----结束

步骤四:在 CAE 环境添加目的访问网络地址

- 步骤1 进入云应用引擎控制台。
- 步骤2 在左侧导航栏选择"系统设置"。
- 步骤3 找到"系统网络配置",单击"编辑"。
- **步骤4** 单击"添加目的网络地址",勾选下一跳为NAT网关的目标网络地址,单击"确定",完成访问公网路由配置。
- **步骤5** 再单击"添加目的网络地址",勾选下一跳为企业路由器的目的网络地址,单击"确定",完成访问私网路由配置。

| | | | | | | ~ | | | |
|-------------------|-------|-------|----------|---------|-----------|-------------|------------------|-----------------------|------------------|
| 添加目的网络地址 | | | | | | ^ | | | |
| ■ 目的网络地址 | IP地址数 | 下一跳类型 | 下一跳 | 类型 | 描述 | | | | |
| Local | 6 | Local | Local | 系统 | | | | | |
| 10.0.0/16 | 1 | 企业路由器 | er-878b | 自定义 | | | | | |
| / 100.93.0.0/16 | 1 | NAT网关 | nat-7a60 | 自定义 | | | | | |
| | | | | | | - | | | |
| | | | | 研 | 取得 | Ħ | 子网 | 状态 | 操作 |
| 【 统配置 ⑦ | | | | | | | | | |
| 统: APM 2.0 | | | | | | | | ! | |
| C: Open relemetry | | | | | | | 短于多位 | ₽ 清沃加副署 | |
| 编辑 | | | | | | | E7080 | | |
| | | | | CAE环境访 | 词VPC | | | | |
| | | | | CAE环境使用 | 目所屬VPC的网络 | 络配置, 实 | 观CAE环境内的组件访问其他网络 | 各内 (VPC、IDC) 的服务。 | |
| 通知规则 ⑦ | | 凭据配 | 3 | 环境所屬VP(| C vpc-w | vc 192.16 | 88.0.0/24 | | |
| 知规则数:3条 | | 已配置外 | 毛据配置数: | 环境所雇子网 | cae-s | subnet 19 | 2.168.0.0/24 | | |
| | | | | 目的网络地址 | 上 提示: | :目的网络均 | 也址是CAE环境所屋子网所关联的 | 9路由表里的目的地址,CAE环境需显示 | 配置要访问的目的地址,才能 |
| 编辑 | | | | | 请求2 | 发送到目的 | 网络,从而实现环境内组件访问第 | w他网络内(包含VPC、自建IDC等)的別 | 股务 (如数据库, 后端服务等) |
| | | | | | | v rakau⊟090 | MERICAL. | | |
| | | | | | 目的 | 的网络地址 | | 来源(路由表) | 操 |
| | | | | | 10.0 | 0.0.0/16 | | rtb-vpc-wc | #P |
| | | | | | | | | | |

----结束