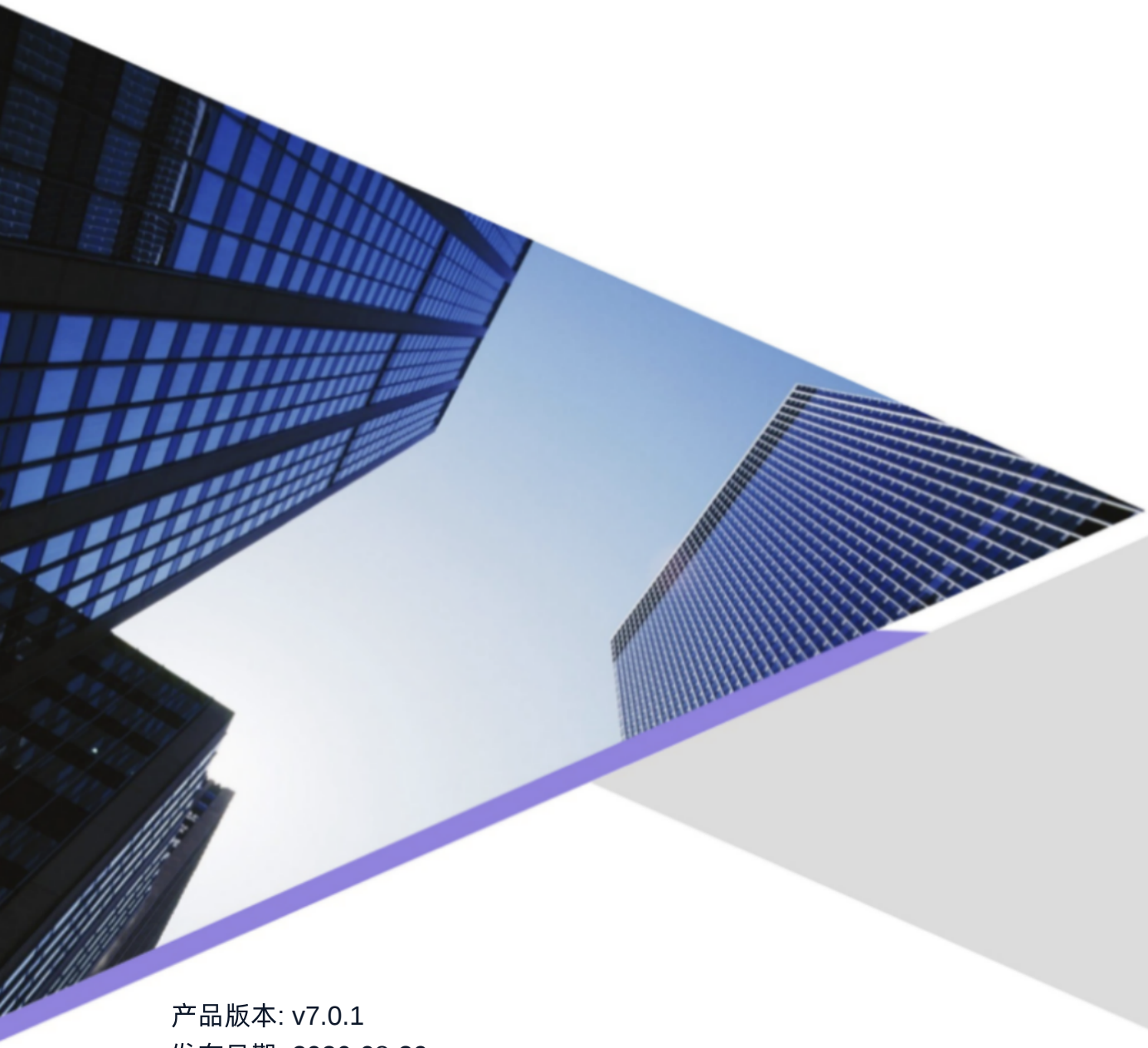
EasyStack
易捷行云

DevOps
最佳实践



产品版本: v7.0.1
发布日期: 2026-08-20

© 北京易捷思达科技发展有限公司

目录

1 最佳实践 ---------- 1
1.1 通过YAML部署自定义构建镜像 ---------- 1
1.2 通过Chart部署自定义构建镜像和Chart模板 ---------- 12
1.3 通过DevOps扩展功能自动构建发布云产品 ---------- 25
1.4 通过DevOps扩展功能部署虚拟化云平台 ---------- 35

版权所有© 北京易捷思达科技发展有限公司

1 最佳实践

1.1 通过YAML部署自定义构建镜像

背景描述

通过DevOps云产品，可以创建从源代码获取到项目构建、测试和部署上线的全流程自动交付流水线，大大缩短交付周期，提升交付效率。本文将以2048游戏应用程序为例，介绍如何快速通过DevOps流水线在构建并发布应用程序镜像后，通过YAML部署该应用程序。

本实践方案中，流水线各项规划信息规划如下：

流水线-YAML部署2048游戏镜像	
Source阶段	项目代码使用 **jack-2048.git** 仓库下的 **master** 分支代码
Build阶段	通过“构建并发布镜像”类型任务构建2048游戏镜像 * Dockerfile文件路径：名为 **Dockerfile** 的文件，其直接放置在代码仓库根目录下，方便任务获取 * 镜像名称：jack-2048 * 镜像版本：alpha-${BUILD_ID} * 工作空间：devops
Deploy阶段	通过“通过YAML部署”类型任务部署2048游戏 * YAML文件：名为2048_env.yaml的文件。其直接放置在代码仓库根目录下，方便任务获取 * 访问域名：2048.example.cn

前提条件

- DevOps流水线需要预先配置承载其运行的Kubernetes集群，具体步骤请参考 配置集群。
- DevOps流水线需要预先关联应用程序源代码的代码仓库，具体步骤请参考 配置代码仓库。

- 本DevOps流水线需要预先制作用于构建镜像的Dockerfile文本文件，并放置在代码仓库的根目录下。本实践方案中，根据规划信息Dockerfile文件的内容如下：

```
FROM hub.ecns.io/devops/jack-2048:latest

COPY 2048 /var/lib/nginx/html/

COPY nginx.conf /etc/nginx/nginx.log.conf

COPY nginx.8880.conf /etc/nginx/nginx.8880.conf

EXPOSE 80

CMD ["nginx", "-c", "/etc/nginx/nginx.8880.conf"]
```

- 本DevOps流水线需要预先制作用于YAML部署的 **2048_env.yaml** 文件，并放置在代码仓库的根目录下。本实践方案中，根据规划信息2048_env.yaml文件的内容如下。其中，**image: hub.ecns.io/devops/jack-2048:alpha-${BUILD_ID}** 参数配置中，**devops** 为流水线“构建”阶段所选择的工作空间，**jack-2048** 为流水线“构建”阶段所构建镜像的名称，**alpha-${BUILD_ID}** 为流水线“构建”阶段所构建镜像的版本。**host: 2048.example.cn** 参数配置中，**2048.example.cn** 为流水线“部署”阶段所规划的访问域名。

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jack-2048
spec:
  selector:
    matchLabels:
      application: jack-2048
  strategy:
    rollingUpdate:
      maxSurge: 3
      maxUnavailable: 70%
    type: RollingUpdate
  template:
    metadata:
      labels:
        application: jack-2048
```

```
    spec:
      containers:
      - image: hub.ecns.io/devops/jack-2048:alpha-${BUILD_ID}
        imagePullPolicy: IfNotPresent
        name: jack-2048
        ports:
        - containerPort: 80
          protocol: TCP
---
apiVersion: v1
kind: Service
metadata:
  name: svc-2048
  labels:
    application: jack-2048
spec:
  ports:
  - name: port-2048
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    application: jack-2048
  type: ClusterIP
---
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ing-2048
spec:
  rules:
  - host: 2048.example.cn
    http:
      paths:
      - backend:
          serviceName: svc-2048
          servicePort: 80
```

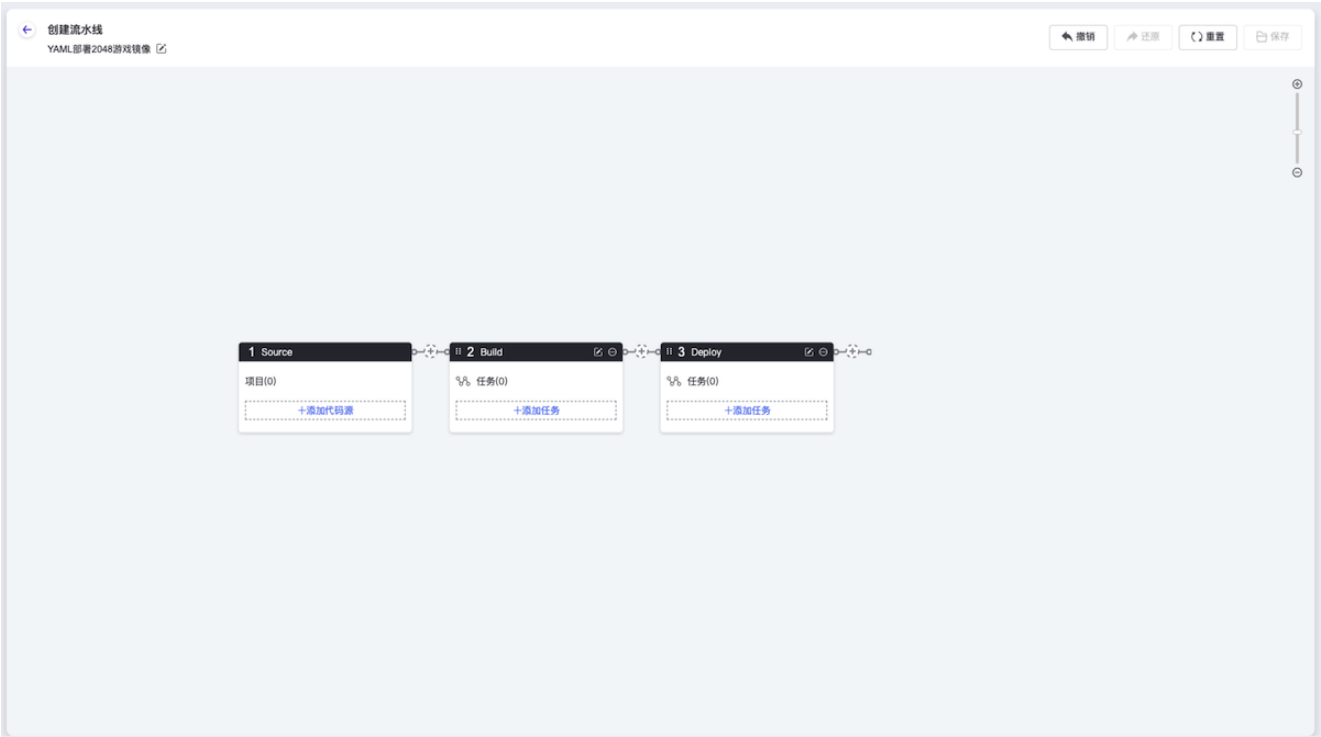
操作步骤

1. 在云平台的顶部导航栏中，依次选择[产品与服务]-[DevOps]-[流水线]，进入“流水线”页面。

2. 在“流水线”页面中，单击页面上方的 `创建流水线` ，弹出“创建流水线”对话框。

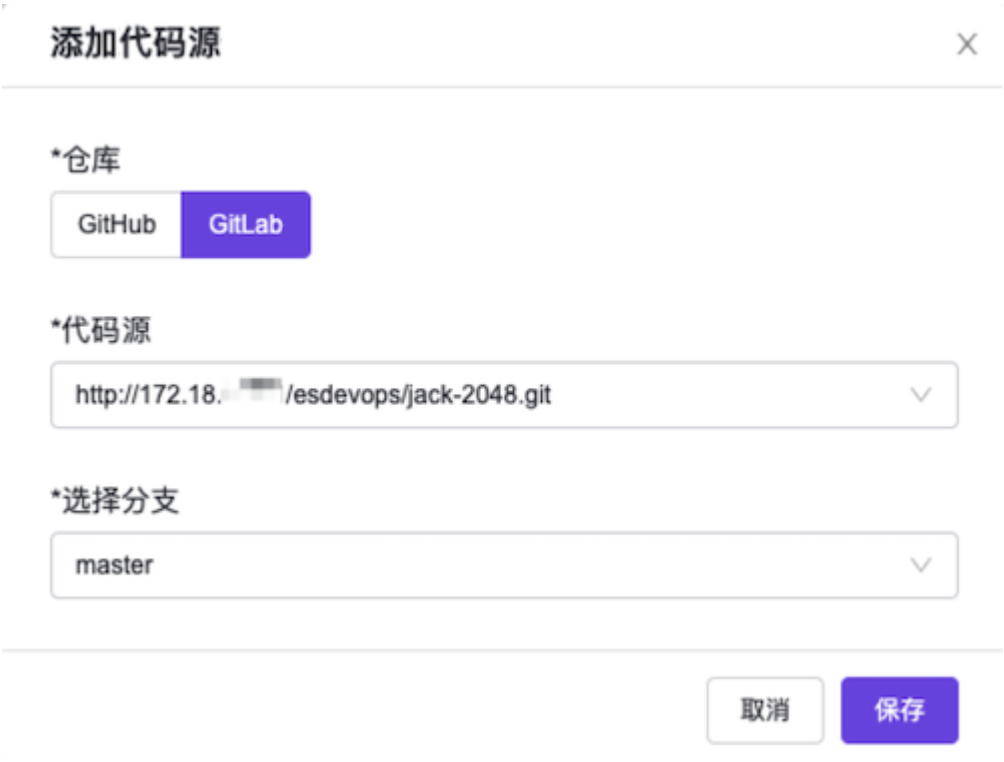
3. 在“创建流水线”对话框中，选择“从零开始创建”后，单击 `创建` ，进入“创建流水线”页面。

4. 在“创建流水线”页面的画布中，根据本次业务需求配置“Source”、“Build”和“Deploy”阶段后，依次在各阶段完成以下任务配置。



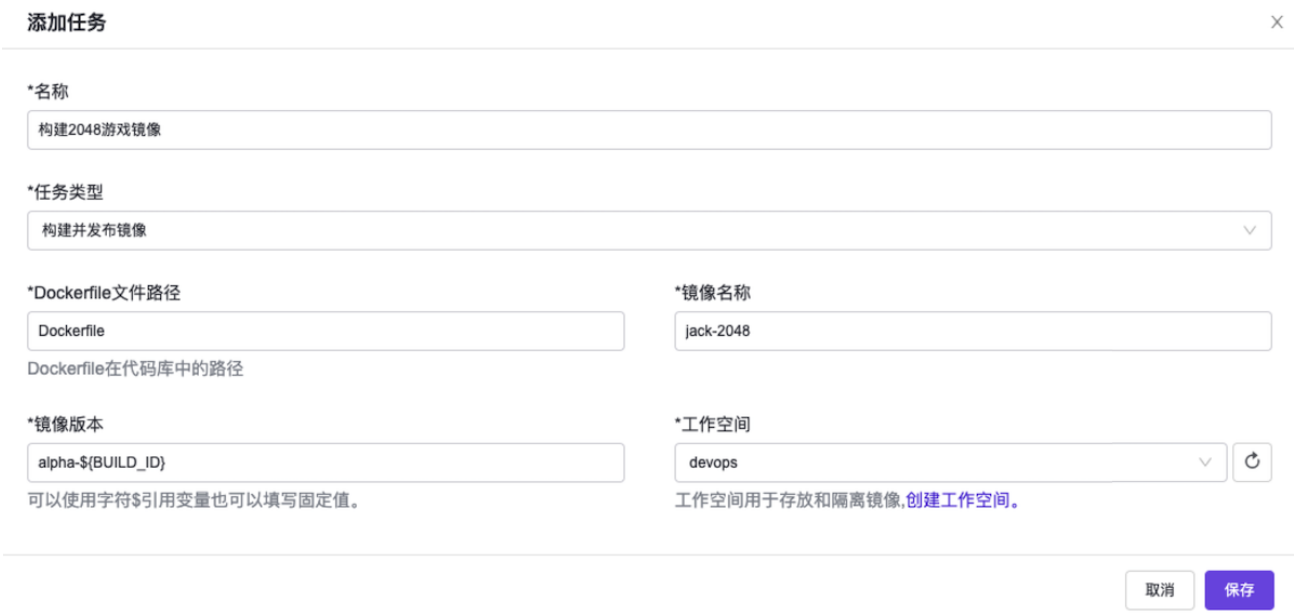
1. 在“Source”阶段添加项目代码源。

   在当前画布的“Source”卡片中，单击``添加代码源``，弹出“添加代码源”对话框。在该对话框中，配置代码源信息后，单击 `保存` ，保存项目的代码源设置，并关闭对话框。



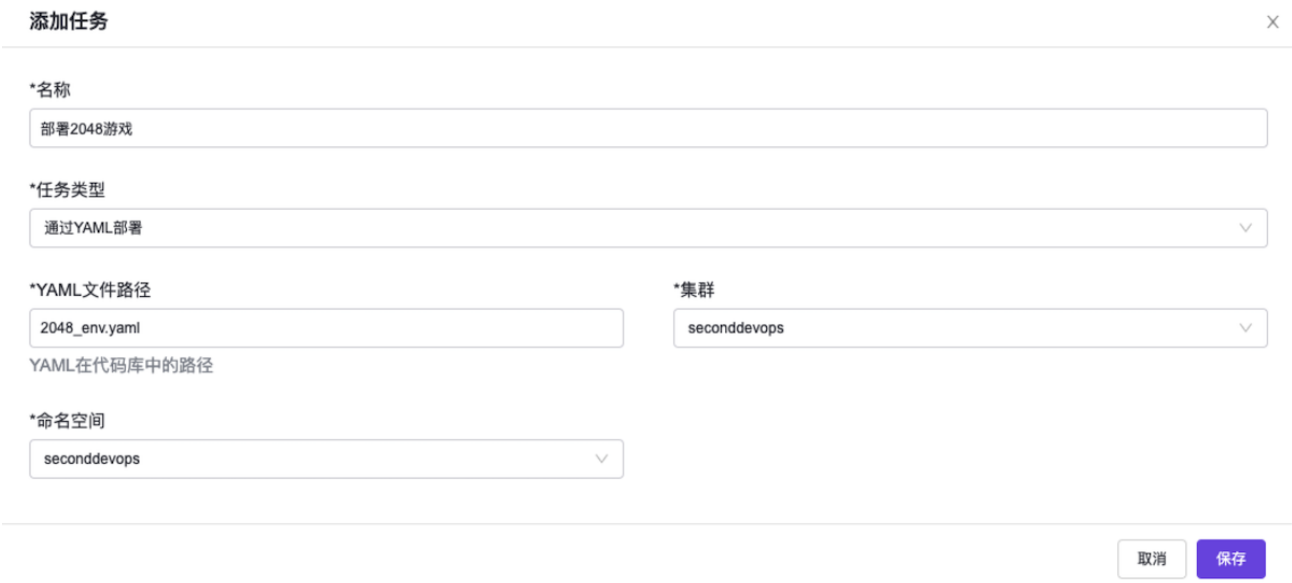
2. 在“Build”阶段添加“构建并发布镜像”任务。

在当前画布的“Build”卡片中，单击 添加任务 ，弹出“添加任务”对话框。在该对话框中，“任务类型”选择“构建并发布镜像”，“Dockerfile文件路径”输入“Dockerfile”，“镜像名称”输入“jack-2048”，“镜像版本”输入“alpha-${BUILD_ID}”，“工作空间”选择“devops”，并配置名称后，单击 保存 ，完成任务创建，并关闭对话框。



3. 在“Deploy”阶段添加“通过YAML部署”任务。

   在当前画布的“Deploy”卡片中，单击 `添加任务` ，弹出“添加任务”对话框。在该对话框中，“任务类型”选择“通过YAML部署”，“YAML路径”输入“2048_env.yaml”，并配置名称和应用程序的部署目标集群及命名空间后，单击 `保存` ，完成任务创建，并关闭对话框。



5. 在“创建流水线”页面的画布中，单击画布右上方的 `保存` 后，在弹出的“保存”对话框中，选择保存方式后，单击 `保存` ，完成流水线创建，并关闭当前页面。

6. 执行流水线。

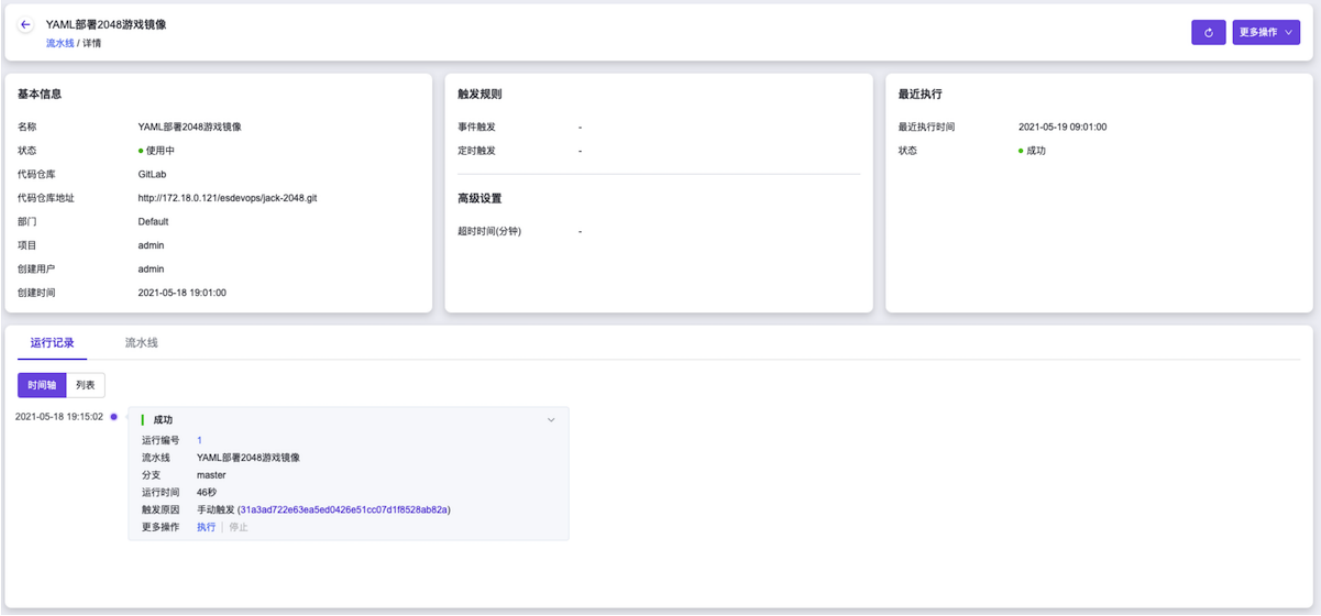
   本实践方案中以手动触发方式为例，触发流水线执行。如需配置流水线自动触发，请参考 [配置流水线执行策略（可选）]。

   1. 在“流水线”页面中，单击上述流水线所在行的 `执行` ，弹出“执行流水线”提示框。
   2. 在“执行流水线”提示框中，单击 `执行` ，执行该流水线，并关闭提示框。

结果验证

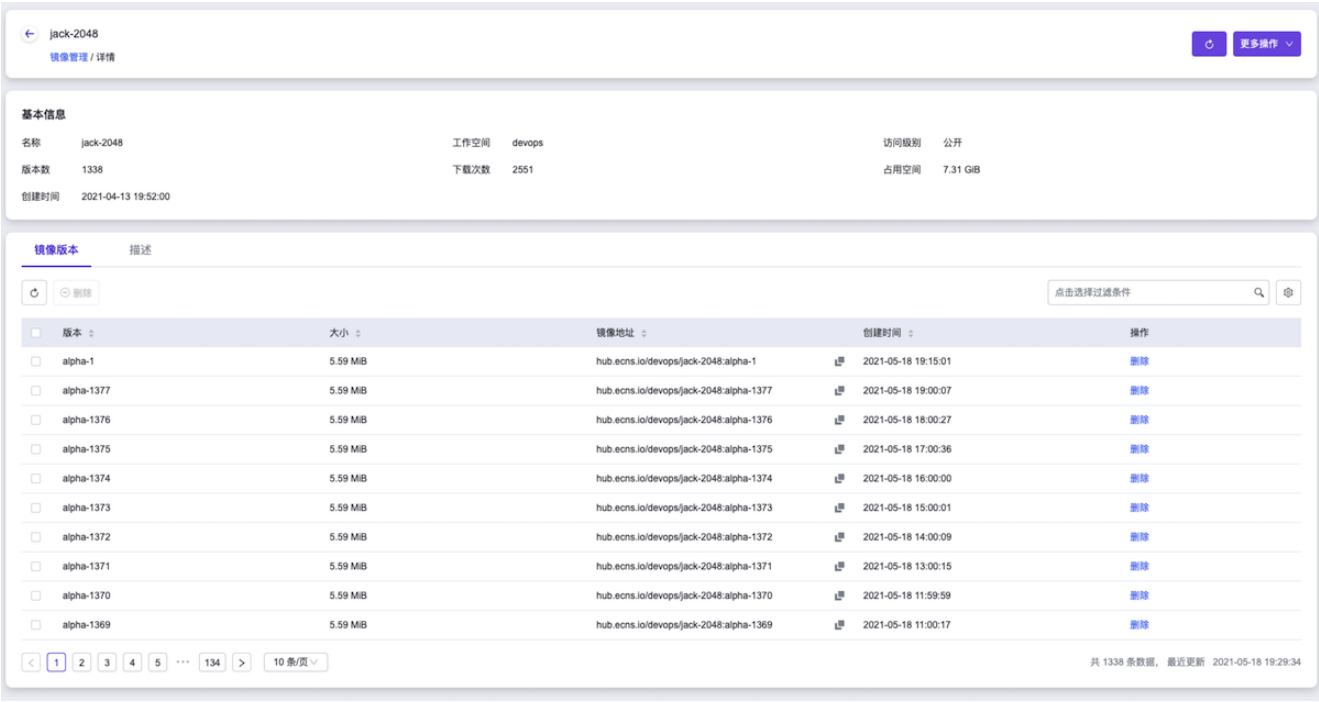
1. 确认流水线成功执行。

   在“流水线”页面中，单击上述流水线名称 ，进入其详情页面。在详情页面的[运行记录]页签中，确认该流水线执行成功并记录此次“运行编号”。



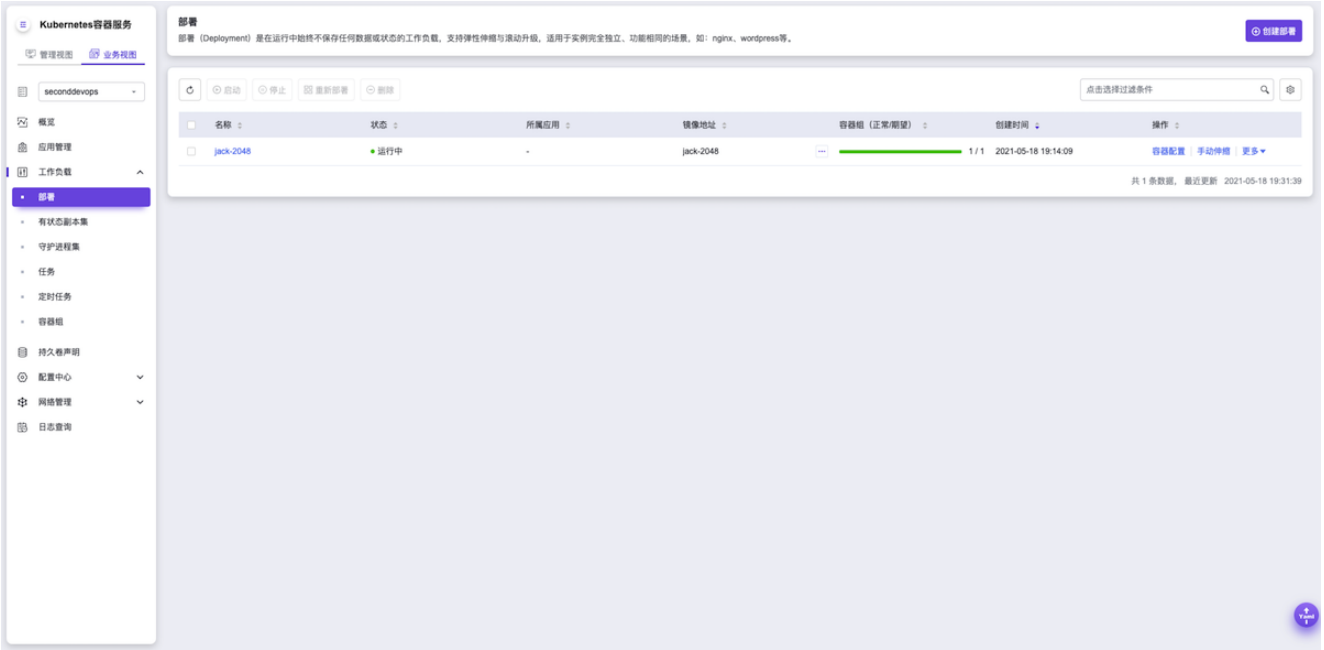
2. 确认镜像成功构建。

   在云平台的顶部导航栏中，依次选择[产品与服务]-[容器服务]-[容器镜像服务]，进入“镜像管理”页面。在该页面中，确认已生成名为“jack-2048”的镜像文件。然后，单击此镜像文件名称，进入其详情页面后，在[镜像版本]页签中，确认该镜像文件的生成版本为“alpha-<运行编号>”。



3. 确认应用程序成功部署。

在云平台的顶部导航栏中，依次选择[产品与服务]-[容器服务]-[Kubernetes容器服务]，进入Kubernetes容器服务页面后，再在该页面的左侧导航栏中，先选择“业务视图”以及该应用程序的所在项目、部署集群和命名空间，再选择[工作负载]-[部署]，进入部署页面。在该页面中，确认已生成名为“jack-2048”的部署且状态为“运行中”。



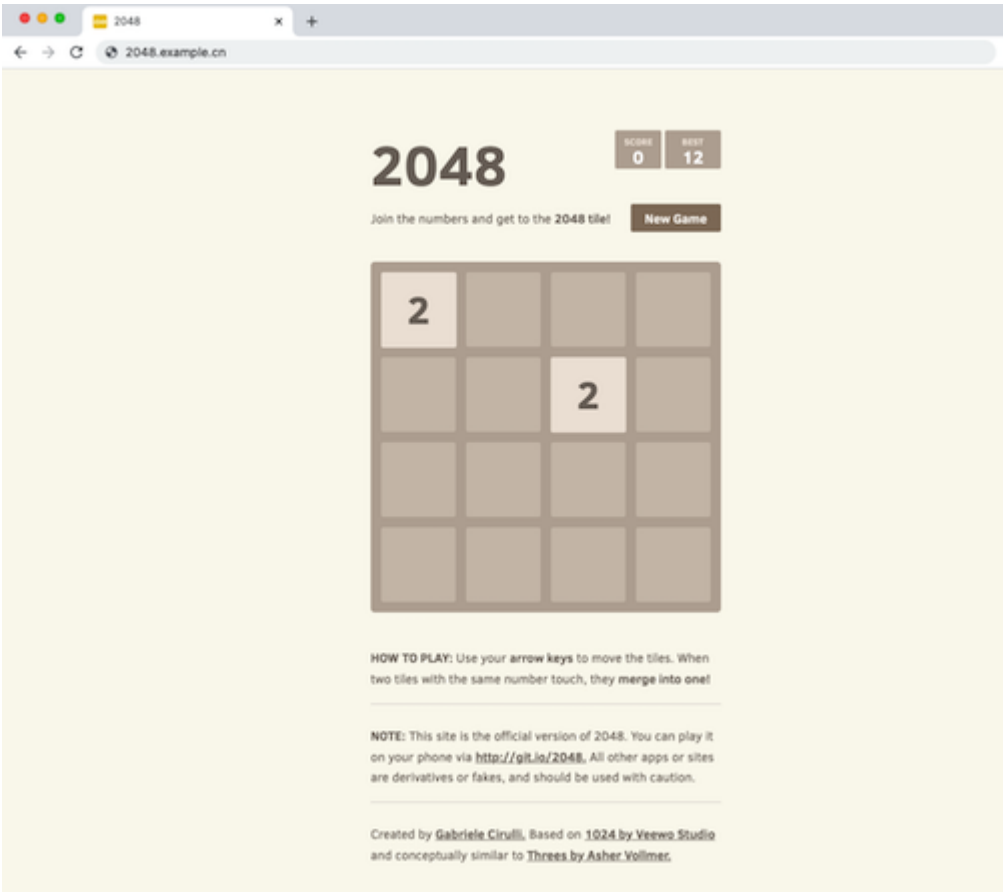
4. 确认应用程序成功访问。

   1. 在云平台的顶部导航栏中，依次选择[产品与服务]-[容器服务]-[Kubernetes容器服务]，进入“管理视图”的“集群管理”页面。在该页面中，查看并记录该应用程序所在Kubernetes集群的IP地址，即该Kubernetes集群所在行中，API Server列https://后的IP地址。

   2. 在本地计算机的 **hosts** 文件中，添加该应用程序所在Kubernetes集群IP地址与YAML文件定义域名的访问映射。

```
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1       localhost
255.255.255.255 broadcasthost
::1             localhost
172.18.(        2048.example.cn
```

   3. 在本地计算机的浏览器地址栏中输入YAML文件定义域名，访问部署的应用程序。


2048
2048.example.cn
2048
SCORE 0
BEST 12
Join the numbers and get to the 2048 tile!
New Game
2
2
HOW TO PLAY: Use your arrow keys to move the tiles. When two tiles with the same number touch, they merge into one!
NOTE: This site is the official version of 2048. You can play it on your phone via http://gitio/2048. All other apps or sites are derivatives or fakes, and should be used with caution.
Created by Gabriele Cirulli. Based on 1024 by Veewo Studio and conceptually similar to Threes by Asher Vollmer.

1.2 通过Chart部署自定义构建镜像和Chart模板

背景描述

通过DevOps云产品，可以创建从源代码获取到项目构建、测试和部署上线的全流程自动交付流水线，大大缩短交付周期，提升交付效率。本文将以nginx应用程序为例，介绍如何快速通过DevOps流水线在构建并发布应用程序镜像后，先打包发布Chart模板，再通过此Chart模板部署该应用程序。

本实践方案中，流水线各项规划信息规划如下：

流水线-Chart部署nginx镜像和Chart模板	
Source阶段	项目代码使用 **nginx.git** 仓库下的 **master** 分支代码
Build阶段	通过“构建并发布镜像”类型任务构建nginx镜像 * Dockerfile文件路径：名为Dockerfile的文件，其直接放置在代码仓库根目录下，方便任务获取 * 镜像名称：nginx * 镜像版本：6.0.1-alpha.${BUILD_ID} * 工作空间：devops
Publish阶段	通过“构建并发布Chart模板”类型任务发布nginx模板 * Chart目录：nginx。即名为Chart.yaml的文件放置在代码仓库的nginx目录下 * 应用模板名称：nginx * 模板版本：6.0.1-alpha.${BUILD_ID}
Deploy阶段	通过“通过Chart模板部署”类型任务部署nginx应用程序 * 部署实例名称：nginx * 应用模板名称：nginx（与发布阶段的应用模板名称保持一致） * 模板版本：6.0.1-alpha.${BUILD_ID}（与发布阶段的模板版本保持一致） * 访问域名：nginx.example.cn

前提条件

- DevOps流水线需要预先配置承载其运行的Kubernetes集群，具体步骤请参考 配置集群。

- DevOps流水线需要预先关联应用程序源代码的代码仓库，具体步骤请参考 配置代码仓库。
- 本DevOps流水线需要预先制作用于构建镜像的Dockerfile文本文件，并放置在代码仓库的根目录下。本实践方案中，根据规划信息Dockerfile文件的内容如下：

```
FROM alpine:latest

RUN sed -i 's/dl-cdn.alpinelinux.org/mirrors.tuna.tsinghua.edu.cn/g'
/etc/apk/repositories \
    && apk --update add nginx \
    && mkdir -p /run/nginx

ADD default.conf /etc/nginx/http.d/
ADD source/html /var/lib/nginx/html/

EXPOSE 80

ENTRYPOINT [ "nginx", "-g", "daemon off;" ]
```

- 本DevOps流水线需要预先制作用于发布Chart模板的 **values.yaml** 文件，并放置在代码仓库的nginx目录下。本实践方案中，根据规划信息values.yaml文件的内容如下。其中，**repository: hub.ecns.io/devops/nginx** 参数配置中，**devops** 为流水线“构建”阶段所选择的工作空间，**nginx** 为流水线“构建”阶段所构建镜像的名称。**tag: 6.0.1-alpha.${BUILD_ID}** 参数配置中，**6.0.1-alpha.${BUILD_ID}** 为流水线“构建”阶段所构建镜像的版本。**host: nginx.example.cn** 参数配置中，**nginx.example.cn** 为流水线“部署”阶段所规划的访问域名。

```
# Default values for nginx-chart.
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.

replicaCount: 1

image:
  repository: hub.ecns.io/devops/nginx
  pullPolicy: IfNotPresent
  # Overrides the image tag whose default is the chart appVersion.
  tag: 6.0.1-alpha.${BUILD_ID}

imagePullSecrets: []
```

```
nameOverride: ""
fullnameOverride: ""

serviceAccount:
  # Specifies whether a service account should be created
  create: false
  # Annotations to add to the service account
  # annotations: {}
  # The name of the service account to use.
  # If not set and create is true, a name is generated using the fullname
template
  name: ""

podAnnotations: {}

podSecurityContext: {}
  # fsGroup: 2000

securityContext: {}
  # capabilities:
  #   drop:
  #   - ALL
  # readOnlyRootFilesystem: true
  # runAsNonRoot: true
  # runAsUser: 1000

service:
  type: ClusterIP
  port: 80

ingress:
  enabled: true
  annotations: {}
    # kubernetes.io/ingress.class: nginx
    # kubernetes.io/tls-acme: "true"
  hosts:
    - host: nginx.example.cn
      paths:
        - path: /

  #  - secretName: chart-example-tls
```

```
  #    hosts:
  #      - chart-example.local

resources: {}
  # We usually recommend not to specify default resources and to leave this
as a conscious
  # choice for the user. This also increases chances charts run on
environments with little
  # resources, such as Minikube. If you do want to specify resources,
uncomment the following
  # lines, adjust them as necessary, and remove the curly braces after
'resources:'.
  # limits:
  #   cpu: 100m
  #   memory: 128Mi
  # requests:
  #   cpu: 100m
  #   memory: 128Mi

autoscaling:
  enabled: false
  minReplicas: 1
  maxReplicas: 100
  targetCPUUtilizationPercentage: 80
  # targetMemoryUtilizationPercentage: 80

nodeSelector: {}

tolerations: []

affinity: {}
```

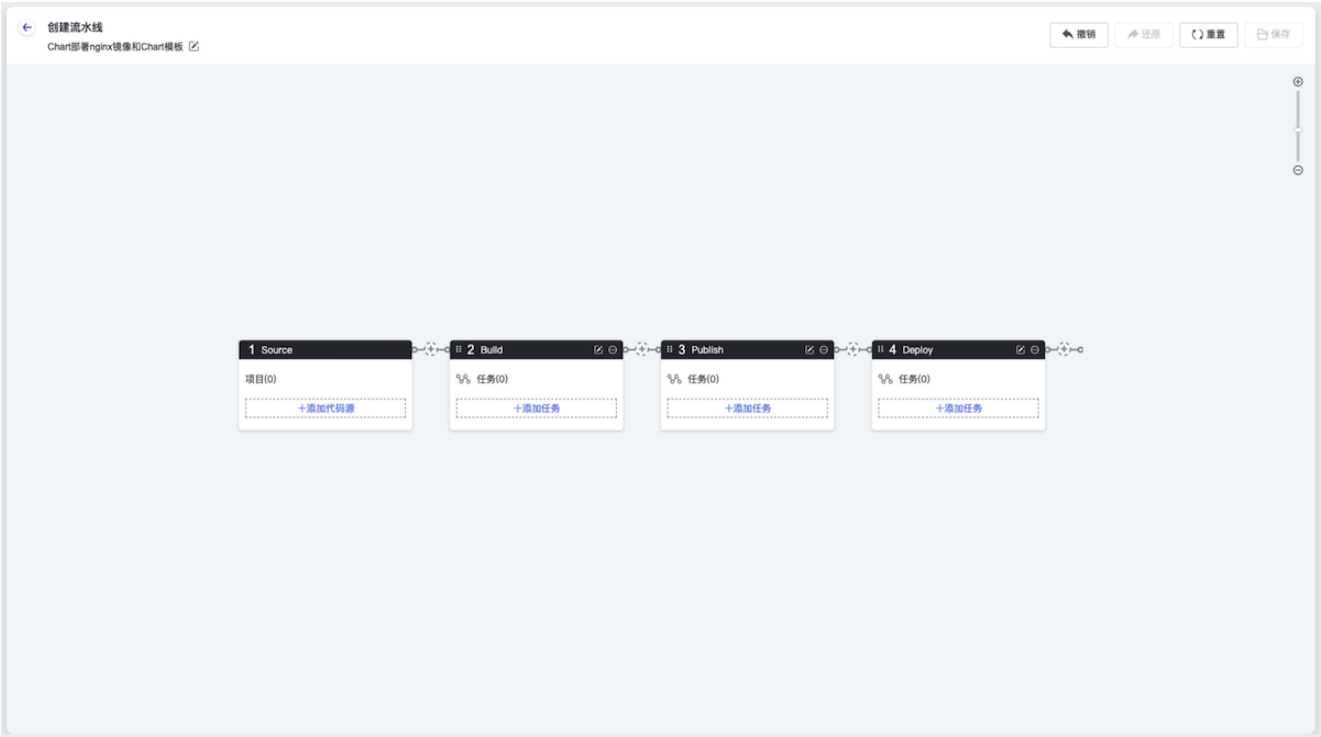
操作步骤

1. 在云平台的顶部导航栏中，依次选择[产品与服务]-[DevOps]-[流水线]，进入“流水线”页面。

2. 在“流水线”页面中，单击页面上方的 `创建流水线` ，弹出“创建流水线”对话框。

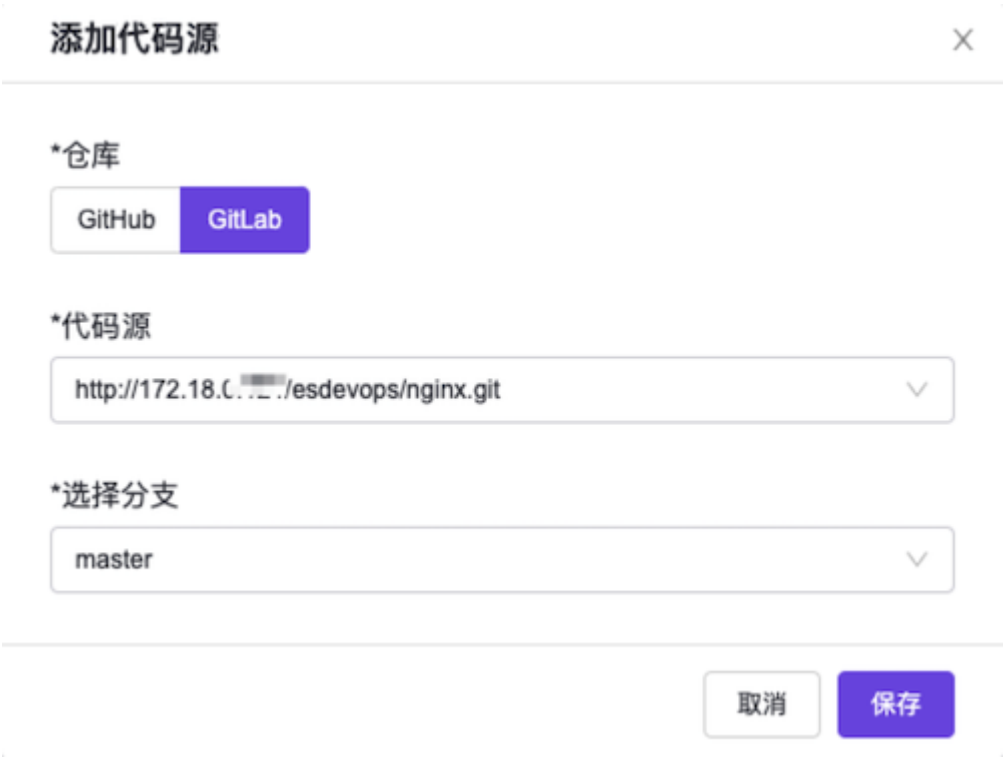
3. 在“创建流水线”对话框中，选择“从零开始创建”后，单击 `创建` ，进入“创建流水线”页面。

4. 在“创建流水线”页面的画布中，根据本次业务需求配置“Source”、“Build”、“Publish”和“Deploy”阶段后，依次在各阶段完成以下任务配置。



1. 在“Source”阶段添加项目代码源。

   在当前画布的“Source”卡片中，单击``添加代码源``，弹出“添加代码源”对话框。在该对话框中，配置代码源信息后，单击 `保存` ，保存项目的代码源设置，并关闭对话框。



2. 在“Build”阶段添加“构建并发布镜像”任务。

在当前画布的“Build”卡片中，单击 添加任务 ，弹出“添加任务”对话框。在该对话框中，“任务类型”选择“构建并发布镜像”，“Dockerfile”输入“Dockerfile”，“镜像名称”输入“nginx”，“镜像版本”输入“6.0.1-alpha.${BUILD_ID}”，“工作空间”选择“devops”，并配置名称后，单击 保存 ，完成任务创建，并关闭对话框。

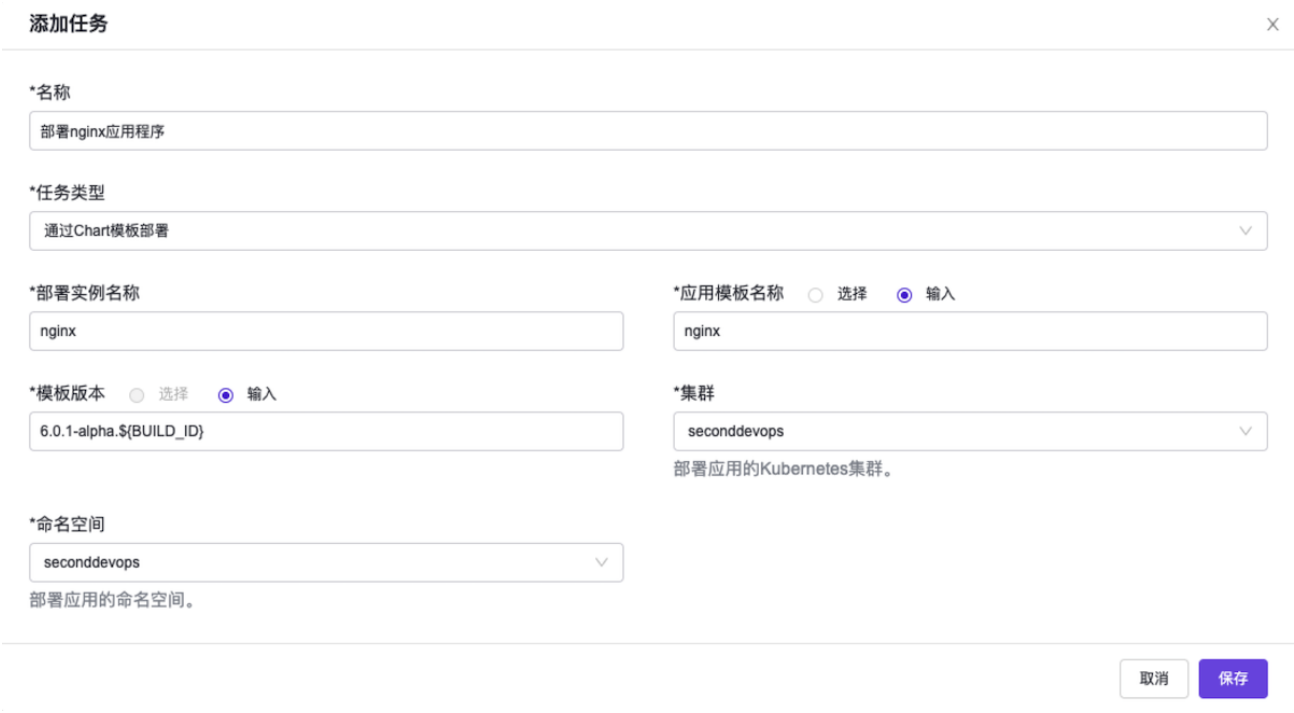
3. 在“Publish”阶段添加“构建并发布Chart模板”任务。

   在当前画布的“Publish”卡片中，单击 `添加任务` ，弹出“添加任务”对话框。在该对话框中，“任务类型”选择“构建并发布Chart模板”，“Chart目录”输入“nginx”，“应用模板名称”输入“nginx”，“模板版本”输入“6.0.1-alpha.${BUILD_ID}”，并配置名称后，单击 `保存` ，完成任务创建，并关闭对话框。

4. 在“Deploy”阶段添加“通过Chart模板部署”任务。

   在当前画布的“Deploy”卡片中，单击 `添加任务` ，弹出“添加任务”对话框。在该对话框中，“任务类型”选择“通过Chart模板部署”，“部署实例名称”输入“nginx”，“应用模板名称”输入“nginx”，“模板版本”输入“6.0.1-alpha.${BUILD_ID}”，并配置名称和Chart模板的部署目标集群及命名空间后，单击 `保存` ，完成任务创建，并关闭对话框。

> 说明：
> 在上述对话框中，“应用模板名称”和“模板版本”支持“选择”和“输入”两种配置方式，本实践方案需要选择“输入”配置方式。关于两种配置方式的具体说明如下，在实际使用过程中请酌情选择：
> - 当待部署的应用模板为“容器应用中心”云产品中已有的自有模板时，请选择“选择”配置方式，依次选择本流水线任务所需的应用模板名称和版本。
> - 当待部署的应用模板为本流水线前述阶段新构建发布的Chart模板时，请选择“输入”配置方式，依次输入前述构建发布阶段所配置的Chart模板的名称和版本。



5. 在“创建流水线”页面的画布中，单击画布右上方的 `保存` 后，在弹出的“保存”对话框中，选择保存方式后，单击 `保存` ，完成流水线创建，并关闭当前页面。

6. 执行流水线。

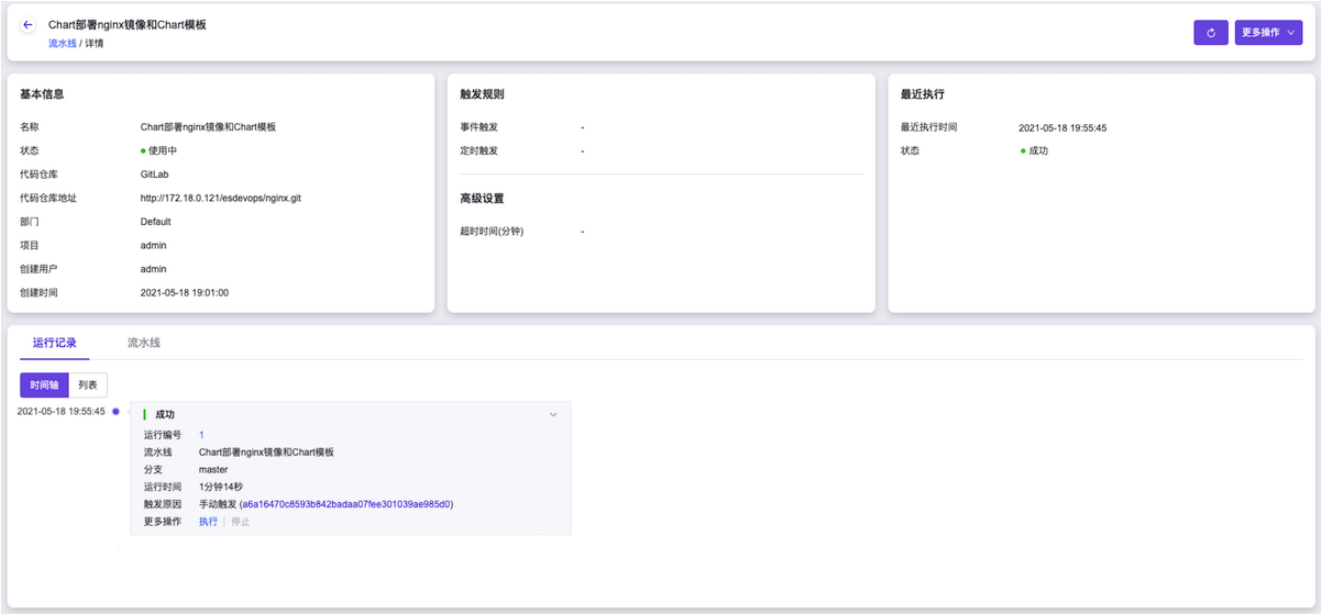
   本实践方案中以手动触发方式为例，触发流水线执行。如需配置流水线自动触发，请参考 [配置流水线执行策略（可选）]。

   1. 在“流水线”页面中，单击上述流水线所在行的 `执行` ，弹出“执行流水线”提示框。
   2. 在“执行流水线”提示框中，单击 `执行` ，执行该流水线，并关闭提示框。

结果验证

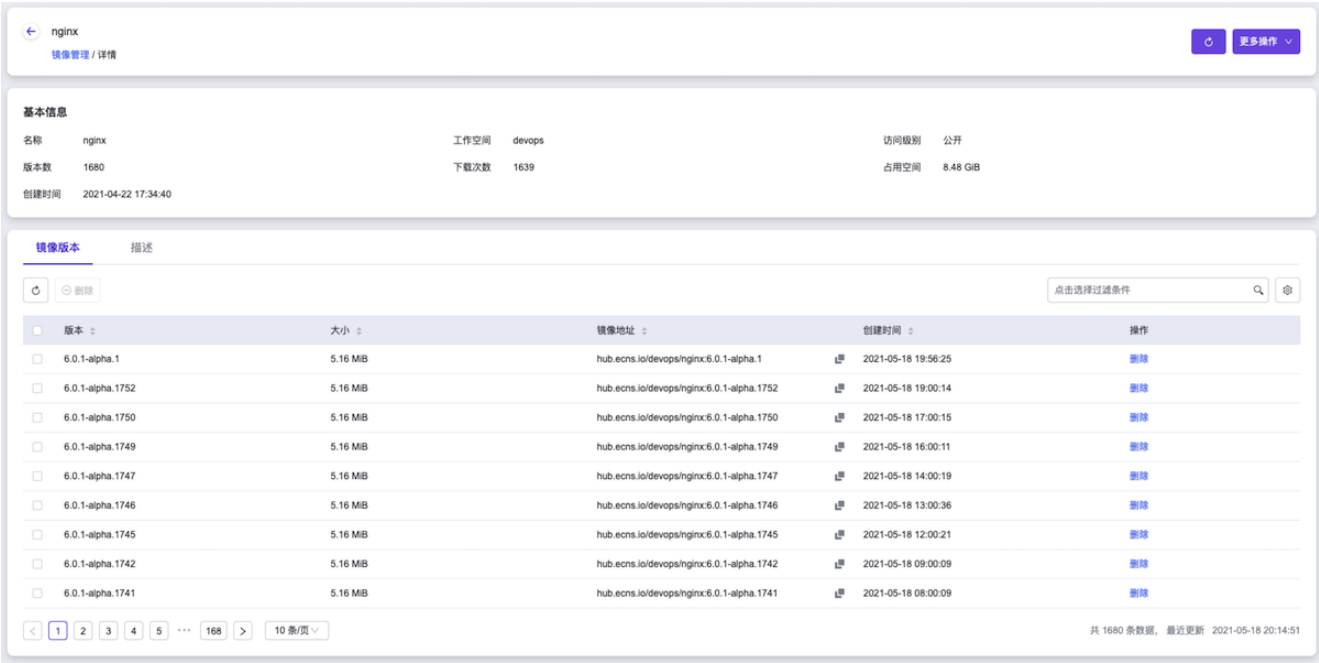
1. 确认流水线成功执行。

   在“流水线”页面中，单击上述流水线名称 ，进入其详情页面。在详情页面的[运行记录]页签中，确认该流水线执行成功并记录此次“运行编号”。



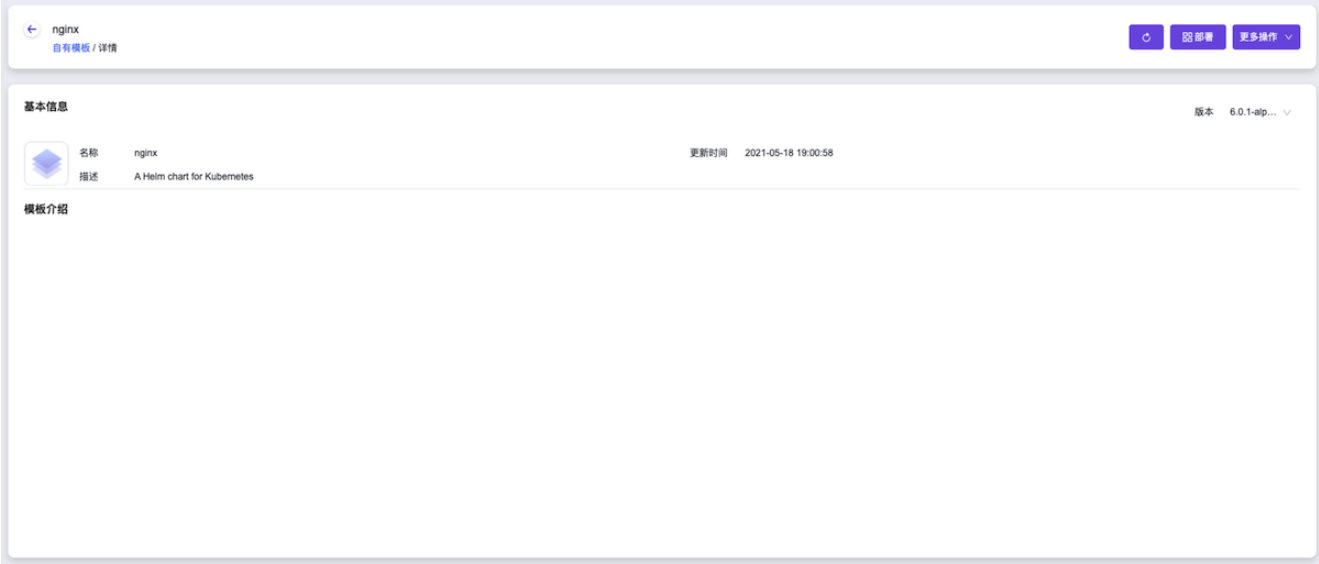
2. 确认镜像成功构建。

   在云平台的顶部导航栏中，依次选择[产品与服务]-[容器服务]-[容器镜像服务]，进入“镜像管理”页面。在该页面中，确认已生成名为“nginx”的镜像文件。然后，单击此镜像文件名称，进入其详情页面后，在[镜像版本]页签中，确认该镜像文件的生成版本为“6.0.1-alpha.<运行编号>”。



3. 确认Chart模板成功发布。

   在云平台的顶部导航栏中，依次选择[产品与服务]-[容器服务]-[容器应用中心]，进入“应用模板”页面。在该页面中，选择[自有模板]页签后，确认已生成名为“nginx”的模板文件。然后，单击此模板文件名称，进入其详情页面后，在“基本信息”区域框的右上方，确认最新生成的该模板文件的版本为“6.0.1-alpha.<运行编号>”。

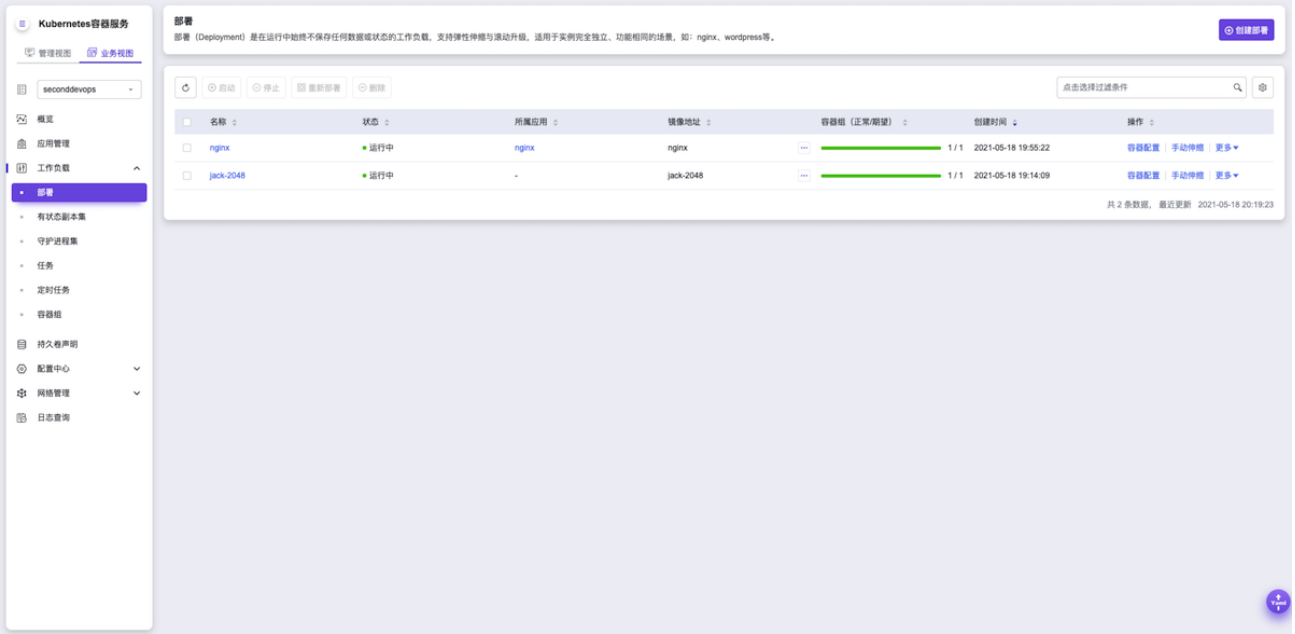


4. 确认应用程序成功部署。

   1. 在云平台的顶部导航栏中，依次选择[产品与服务]-[容器服务]-[容器应用中心]，进入容器应用中心服务页面后，再在该页面的左侧导航栏中，选择“模板实例”，进入“模板实例”页面。在该页面中，确认已生成名

为“nginx”的模板实例，且“模板来源”为“nginx: 6.0.1-alpha.<运行编号>”。然后，单击此模板实例名称，进入其详情页面后，在[工作负载]页签中，确认所用镜像为“hub.ecns.io/devops/nginx:6.0.1-alpha.<运行编号>”。

2. 在云平台的顶部导航栏中，依次选择[产品与服务]-[容器服务]-[Kubernetes容器服务]，进入Kubernetes容器服务页面后，再在该页面的左侧导航栏中，先选择“业务视图”以及该应用程序的所在项目、部署集群和命名空间，再选择[工作负载]-[部署]，进入部署页面。在该页面中，确认已生成名为“nginx”的部署且状态为“运行中”。



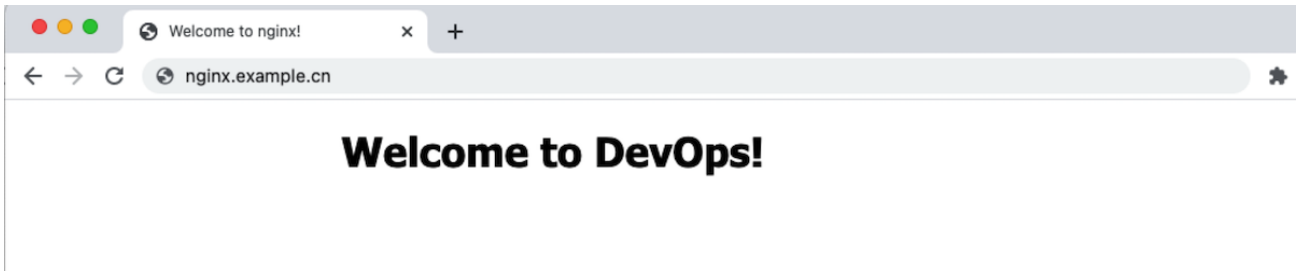
5. 确认应用程序成功访问。

1. 在云平台的顶部导航栏中，依次选择[产品与服务]-[容器服务]-[Kubernetes容器服务]，进入“管理视图”的“集群管理”页面。在该页面中，查看并记录该应用程序所在Kubernetes集群的IP地址，即该Kubernetes集群所在行中，API Server列https://后的IP地址。

2. 在本地计算机的 **hosts** 文件中，添加该应用程序所在Kubernetes集群IP地址与values.yaml文件定义域名的访问映射。

```
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1       localhost
255.255.255.255 broadcasthost
::1             localhost
172.18.[redacted]  2048.example.cn
172.18.[redacted]  nginx.example.cn
```

3. 在本地计算机的浏览器地址栏中输入values.yaml文件定义域名，访问部署的应用程序。



1.3 通过DevOps扩展功能自动构建发布云产品

背景描述

DevOps支持通过将预置好的工具集镜像上传至容器镜像服务云产品后，与流水线的运行脚本类任务相互配合使用，达到扩展DevOps功能的目的。本公司秉承“eat your own dog food & 使用自己的产品设计开发云产品”的设计理念，所以，本公司云平台中所有云产品的发布均是通过DevOps扩展功能支撑实现的，其中不乏外部开发者开发的云产品，如恩耐博慧海数据平台和安全狗云眼平台等。本文将以此为例，介绍如何快速通过DevOps扩展功能，实现云产品的自动化构建并同步发布在OTA服务器中。

本实践方案中，流水线各项规划信息规划如下：

流水线-DevOps扩展功能自动构建发布云产品	
基本信息	* 云产品版本号：6.0.1-alpha.${BUILD_ID} * 本流水线各阶段的所有“运行脚本”类型任务中均使用此工具集镜像文件： * 镜像地址：名为 **docker** 的镜像文件，其作为自有镜像，上传在“容器镜像服务”云产品的工作空间中 * 镜像版本：dind-centos-final-v4
Source阶段	项目代码使用 **build-config.git** 仓库下的 **master** 分支代码
Initial阶段	通过“运行脚本”类型任务设置 **Config** 配置文件中的云产品名称 * 脚本：bash common/replace_config_content.sh * 环境变量：COMPONENTS_TO_BUILD = *云产品名称*

流水线-DevOps扩展功能自动构建发布云产品	
Build阶段	通过“登录Harbor”、“检查OTA服务器metadata版本”、“下载registry-cloud-product全量镜像”和“生成云产品manifest”四个任务获得云产品包构建所需的manifest * 登录Harbor：通过“运行脚本”类型任务登录Harbor仓库 * 脚本： docker login hub.example.cn -u ${dockerHubUser} -p $ {dockerHubPassword} docker login hub.example.io -u ${dockerHubUser} -p $ {dockerHubPassword} * 环境变量： dockerHubUser = *登录用户名* dockerHubPassword = *登录密码* * 检查OTA服务器metadata版本：通过“运行脚本”类型任务检查OTA服务器的metadata版本是否重复 * 脚本： bash common/check_cloud_product_metadata.sh * 下载云产品注册镜像：通过“运行脚本”类型任务下载registry-cloud-product全量镜像文件 * 脚本：（*172.16.XX.XX* 为本公司File Server的IP地址） cd / && wget -q http://172.16.XX.XX:8000/build_cloud_product_file/x86_64/registry-cloud-product.tar tar -zxvf registry-cloud-product.tar > /dev/null * 生成云产品manifest：通过“运行脚本”类型任务生成云产品的manifest * 脚本： bash product/manifest_generator.sh
Deploy阶段	通过“运行脚本”类型任务构建云产品包并发布在OTA服务器中 * 脚本： bash product/ota_file_uploader.sh

前提条件

- DevOps流水线需要预先配置承载其运行的Kubernetes集群，具体步骤请参考 配置集群。
- DevOps流水线需要预先关联应用程序源代码的代码仓库，具体步骤请参考 配置代码仓库。

- 本DevOps流水线需要预先获取名为 **docker** 镜像文件存放在本地计算机中，用于在流水线执行过程中提供工具集镜像。

操作步骤

1. 上传镜像文件。

   本实践方案中以在云平台界面上传的方式为例，上传镜像文件。如需从Docker客户端和Containerd客户端中直接推送镜像，请参考“容器镜像服务”帮助中“上传镜像”的相关内容。

   1. 在云平台的顶部导航栏中，依次选择[产品与服务]-[容器服务]-[容器镜像服务]，进入“镜像管理”页面。

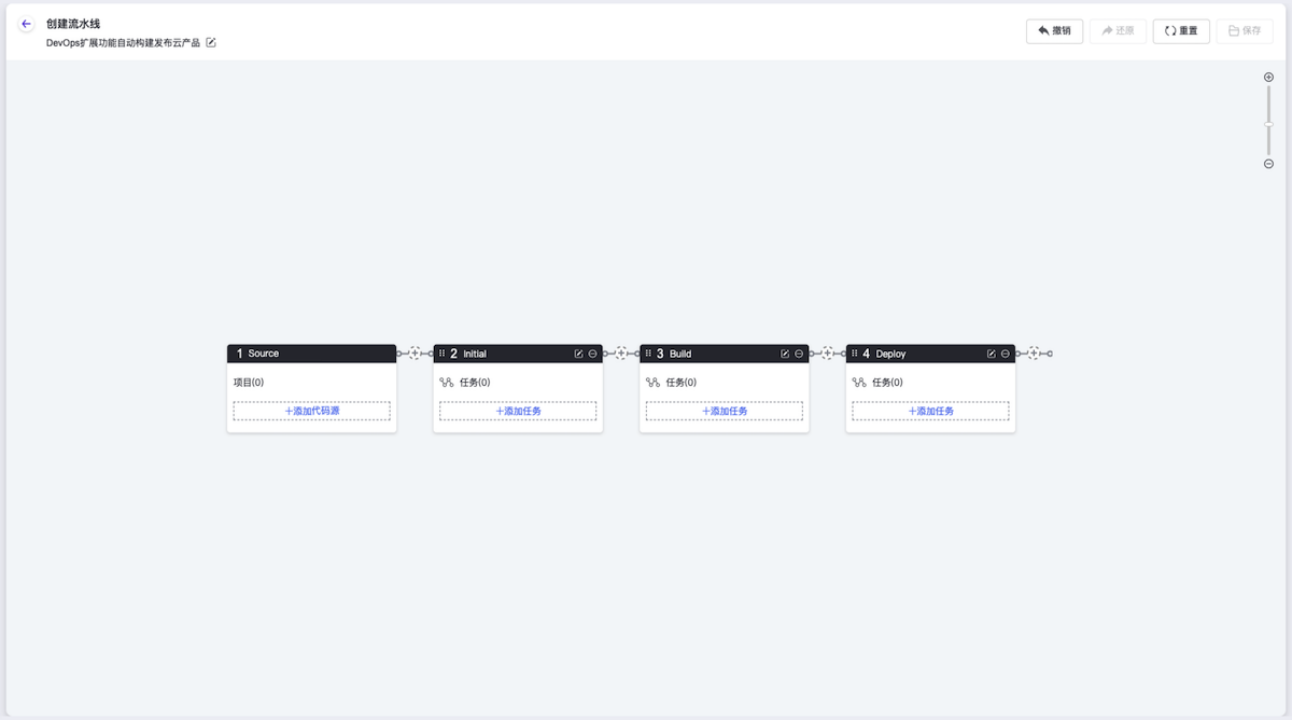
   2. 在“镜像管理”页面中，选择[自有镜像]页签后，单击列表上方的 上传镜像 ，弹出“上传镜像”对话框。

   3. 在“上传镜像”对话框中，选择工作空间和本地镜像文件后，单击 上传 ，开始上传镜像文件，并关闭当前对话框。

2. 创建流水线。

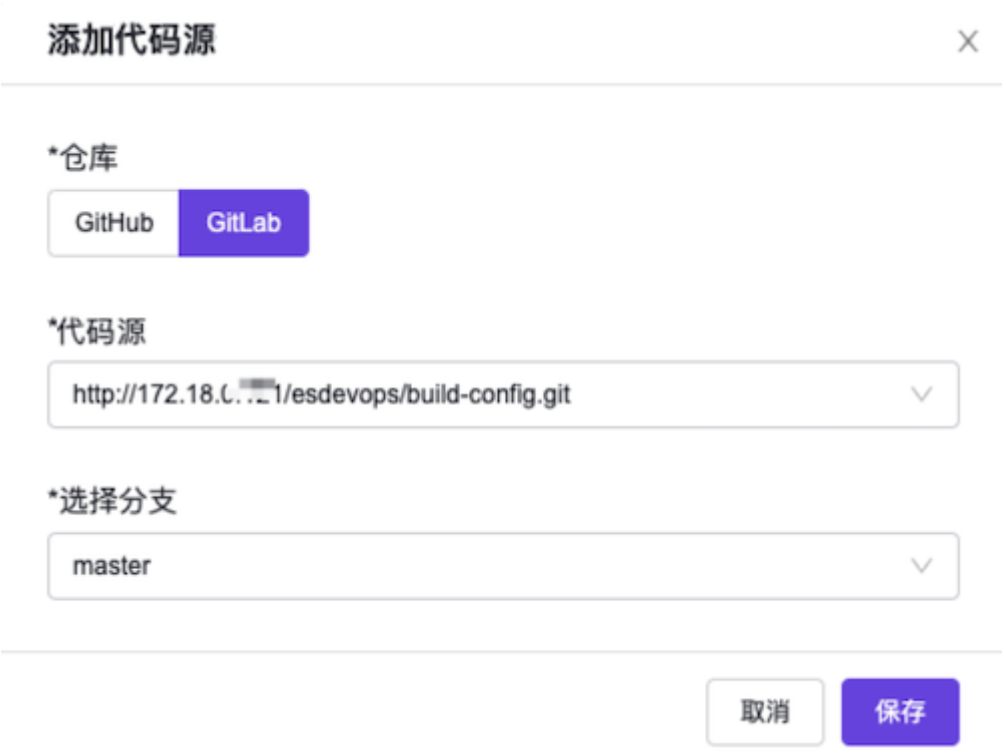
   1. 在云平台的顶部导航栏中，依次选择[产品与服务]-[DevOps]-[流水线]，进入“流水线”页面。
   2. 在“流水线”页面中，单击页面上方的 `创建流水线` ，弹出“创建流水线”对话框。
   3. 在“创建流水线”对话框中，选择“从零开始创建”后，单击 `创建` ，进入“创建流水线”页面。

4. 在“创建流水线”页面的画布中，根据本次业务需求配置“Source”、“Initial”、“Build”和“Deploy”阶段后，依次在各阶段完成对应任务配置。



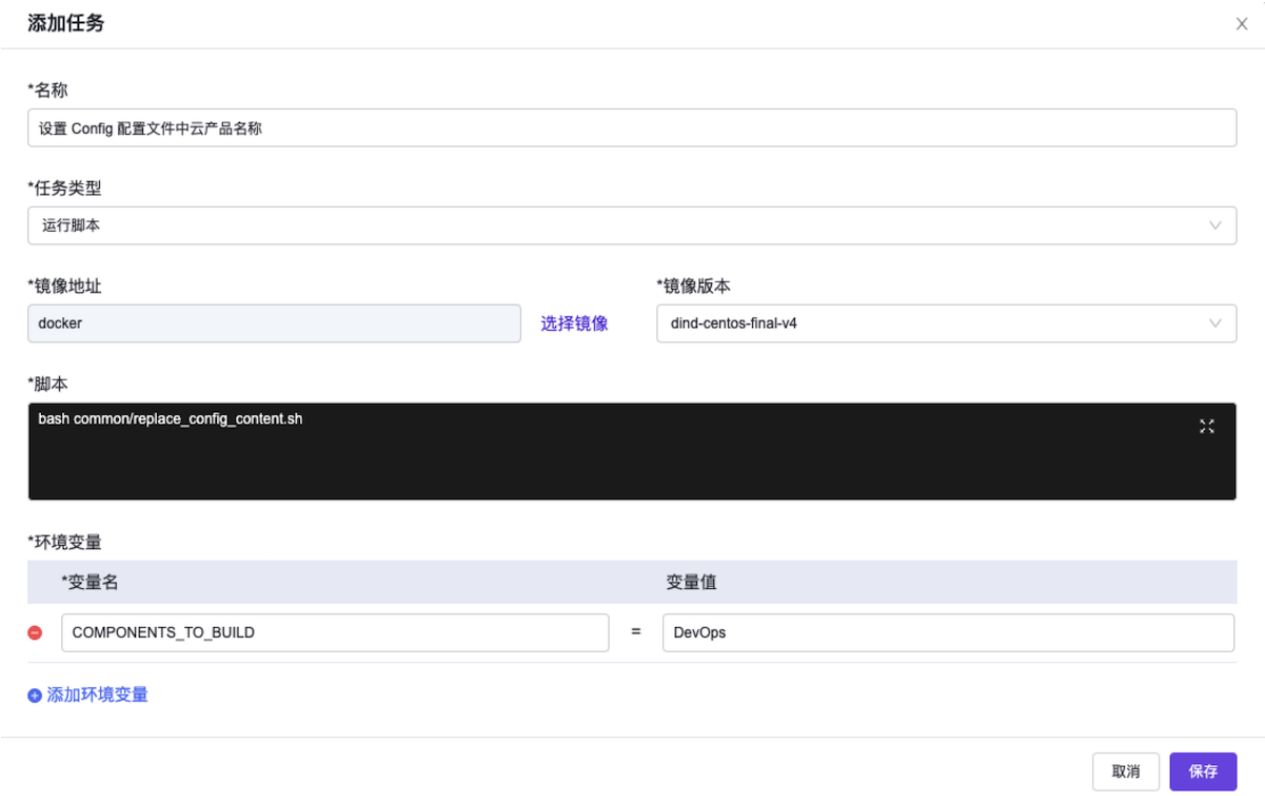
1. 在“Source”阶段添加项目代码源。

   在当前画布的“Source”卡片中，单击``添加代码源``，弹出“添加代码源”对话框。在该对话框中，配置代码源信息后，单击 保存 ，保存项目的代码源设置，并关闭对话框。



2. 分别在“Initial”、“Build”和“Deploy”阶段添加对应的“运行脚本”任务。

   在当前画布的对应阶段卡片中，单击 添加任务 ，弹出“添加任务”对话框。在该对话框中，“任务类型”选择“运行脚本”，“镜像地址”选择[自有镜像]页签中的“docker”，“镜像版本”选择“dind-centos-final-v4”，并配置名称及对应方案规划信息中的脚本内容和环境变量后，单击 保存 ，完成任务创建，并关闭对话框。



5. 在“创建流水线”页面的画布中，单击画布右上方的 保存 后，在弹出的“保存”对话框中，选择保存方式后，单击 保存 ，完成流水线创建，并关闭当前页面。

3. 执行流水线。

本实践方案中以手动触发方式为例，触发流水线执行。如需配置流水线自动触发，请参考 [配置流水线执行策略（可选）]。

1. 在“流水线”页面中，单击上述流水线所在行的 `执行` ，弹出“执行流水线”提示框。
2. 在“执行流水线”提示框中，单击 `执行` ，执行该流水线，并关闭提示框。

结果验证

1. 确认流水线成功执行。

   在“流水线”页面中，单击上述流水线名称 ，进入其详情页面。在详情页面的[运行记录]页签中，确认该流水线执行成功并记录此次“运行编号”。

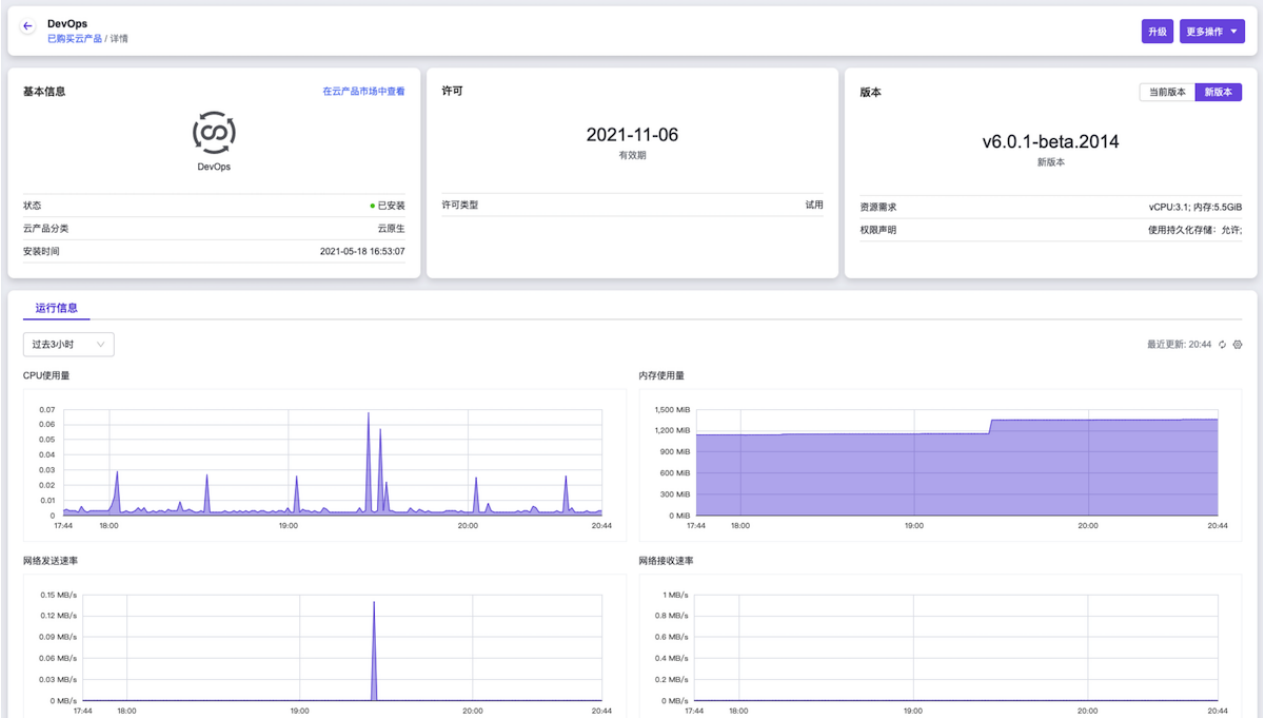
2. 确认云产品成功发布。

   当该云产品在云平台已安装时，请直接执行下述操作。当该云产品还未在云平台安装时，请先在云产品市场中获取该云产品，再执行下述操作。

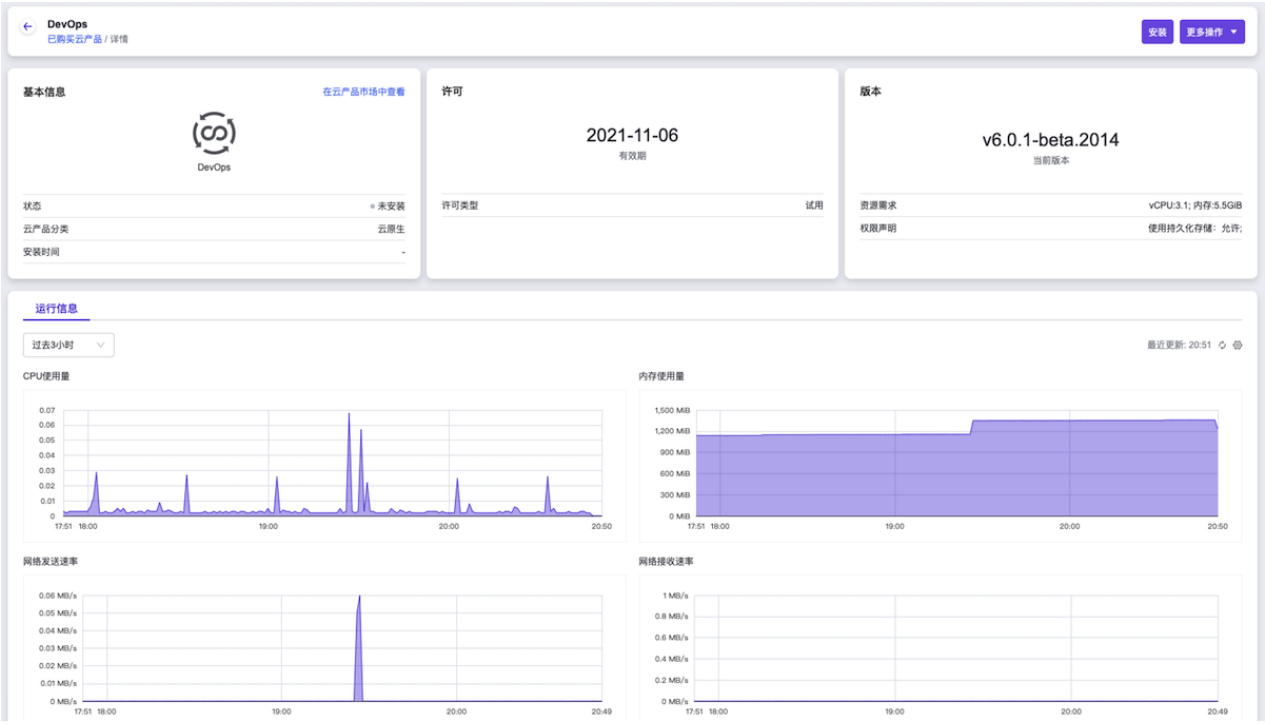
   1. 在云平台的顶部导航栏中，依次选择[产品与服务]-[产品与服务管理]-[已购买云产品]，进入“已购买云产品”页面。
   2. 在“已购买云产品”页面中，根据该云产品安装状态，酌情选择下述对应操作确认该云产品成功发布。

- 当该云产品为已安装状态时，请首先在该页面中确认，该云产品所在行的“已安装版本”后有“可升级”提示。

然后，在该页面中单击该云产品名称，进入其详情页面。在详情页面的“版本”区域框中，选择[新版本]页签，确认该云产品新版本为“6.0.1-alpha.<运行编号>”。



- 当该云产品为未安装状态时，单击该云产品名称，进入其详情页面。在该页面的“版本”区域框中，确认该云产品版本为“6.0.1-alpha.<运行编号>”。


DevOps
已购买云产品 / 详情
安装
更多操作
基本信息
在云产品市场中查看
DevOps
状态
未安装
云产品分类
云原生
安装时间
-
许可
2021-11-06
有效期
许可类型
试用
版本
v6.0.1-beta.2014
当前版本
资源需求
vCPU:3.1; 内存:5.5GiB
权限声明
使用持久化存储：允许;
运行信息
过去3小时
最近更新: 20:51
CPU使用量
内存使用量
网络发送速率
网络接收速率

1.4 通过DevOps扩展功能部署虚拟化云平台

背景描述

DevOps支持通过将预置好的工具集镜像上传至容器镜像服务云产品后，与流水线的运行脚本类任务相互配合使用，达到扩展DevOps功能的目的。本公司秉承“eat your own dog food & 使用自己的产品设计开发云产品”的设计理念，所以，在日常工作中本公司通过利用云基础设施的资源编排能力和DevOps扩展功能，在一个云平台中编排部署多个虚拟化云平台，用于为工程技术团队提供每天最新的研发或测试环境。本文将以此为例，介绍如何快速通过DevOps扩展功能部署虚拟化云平台。

本实践方案中，流水线各项规划信息规划如下：

流水线-DevOps扩展功能部署虚拟化云平台	
Source阶段	项目代码使用 **os-in-os.git** 仓库下的 **master** 分支代码
Deploy阶段	通过“运行脚本”类型任务部署虚拟化云平台 * 镜像地址：名为 **escloud-linux-source-busybox** 的镜像文件，其作为自有镜像，上传在“容器镜像服务”云产品的工作空间中 * 镜像版本：6.0.1

前提条件

- DevOps流水线需要预先配置承载其运行的Kubernetes集群，具体步骤请参考 配置集群。
- DevOps流水线需要预先关联应用程序源代码的代码仓库，具体步骤请参考 配置代码仓库。
- 本DevOps流水线需要预先获取名为 **escloud-linux-source-busybox** 镜像文件存放在本地计算机中，用于在流水线执行过程中提供工具集镜像。
- 本DevOps流水线需要预先获取名为 **stack-3nodes-3cloud-product-nodes.yaml** 的YAML文件放置在代码仓库的根目录下，用于在流水线执行过程中进行资源编排部署。
- 本DevOps流水线需要预先制作用于检查虚拟化云平台部署状态的 **check_stack_status.sh** 脚本文件，并放置在代码仓库的根目录下。本实践方案中，根据规划信息check_stack_status.sh脚本文件的内容如下：

```
#!/usr/bin/env bash

source ./openrc-sz

if heat stack-list | grep CREATE_IN_PROGRESS | grep $1 ; then
    echo 'Heat stack is creating'
    while true; do
        if heat stack-list | grep CREATE_COMPLETE | grep $1; then
            echo "stack create complete"
            exit
        fi
        if heat stack-list | grep CREATE_FAILED | grep $1; then
            echo "stack create failed"
            exit 1
        fi
        sleep 300
    done
else
    echo 'No stack is creating or stack create failed, so exit with 1'
    exit 1
fi
```

操作步骤

1. 上传镜像文件。

   本实践方案中以在云平台界面上传的方式为例，上传镜像文件。如需从Docker客户端和Containerd客户端中直接推送镜像，请参考“容器镜像服务”帮助中“上传镜像”的相关内容。

   1. 在云平台的顶部导航栏中，依次选择[产品与服务]-[容器服务]-[容器镜像服务]，进入“镜像管理”页面。

   2. 在“镜像管理”页面中，选择“自有镜像”页签后，单击列表上方的 `上传镜像` ，弹出“上传镜像”对话框。

3. 在“上传镜像”对话框中，选择工作空间和本地预先获取的镜像文件后，单击 上传 ，开始上传镜像文件，并关闭当前对话框。

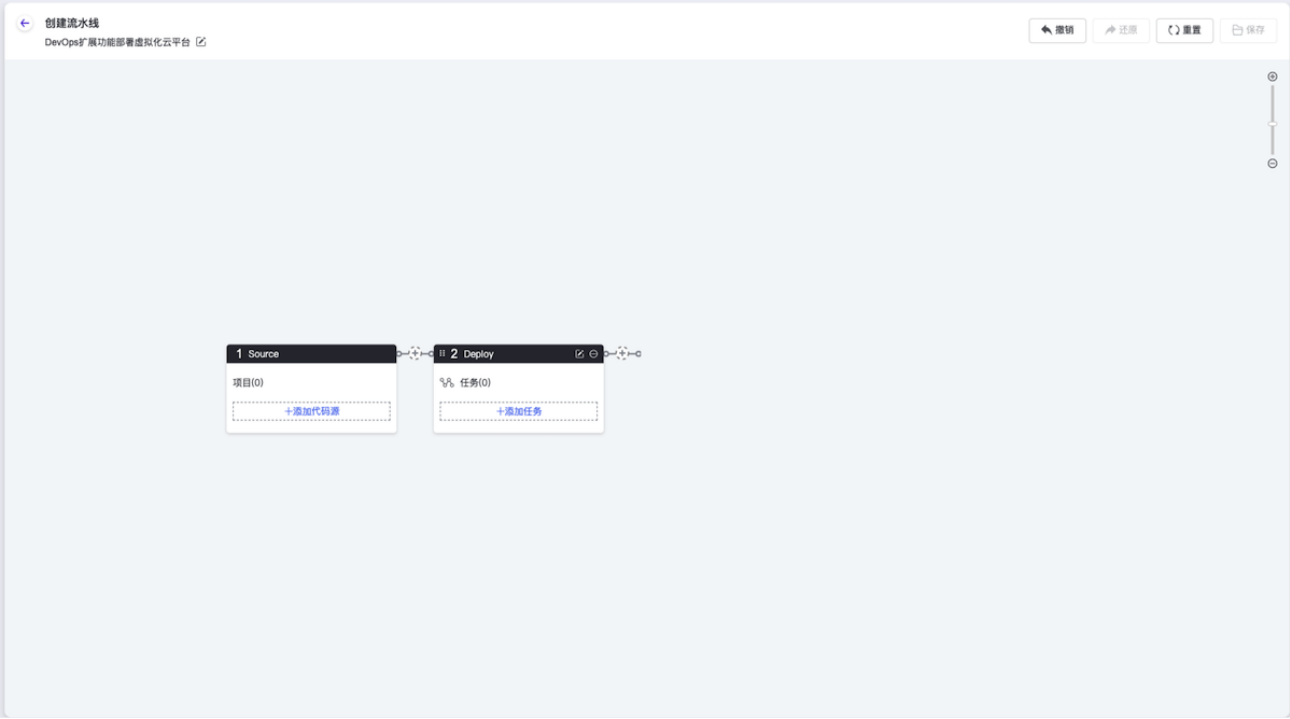
2. 创建流水线。

1. 在云平台的顶部导航栏中，依次选择[产品与服务]-[DevOps]-[流水线]，进入“流水线”页面。

2. 在“流水线”页面中，单击页面上方的 `创建流水线` ，弹出“创建流水线”对话框。

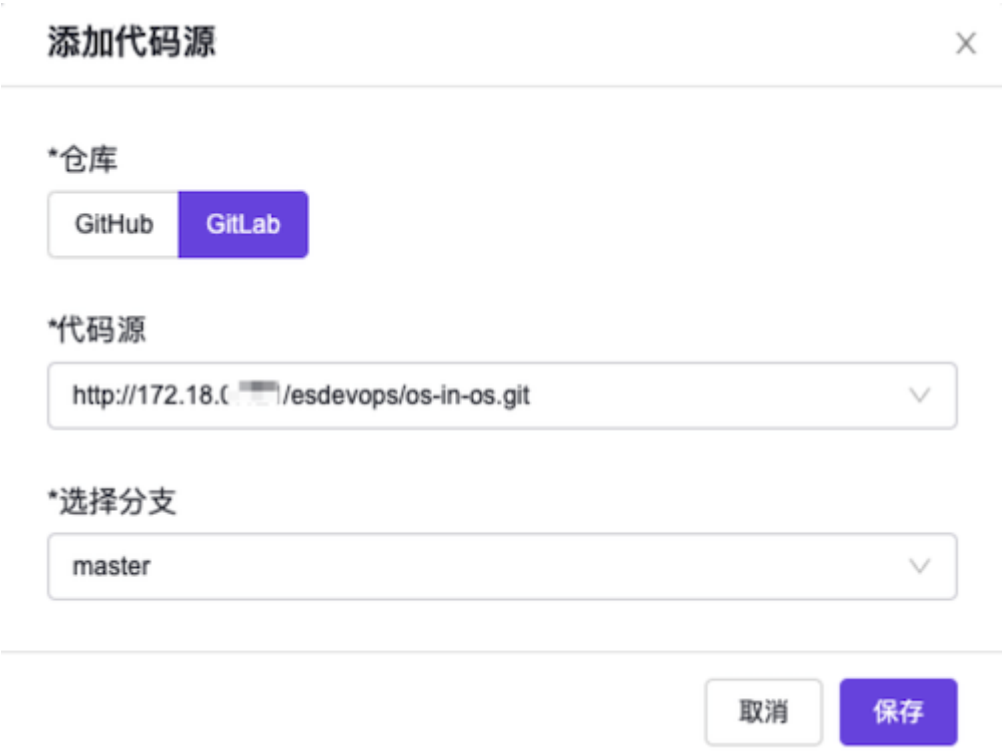
3. 在“创建流水线”对话框中，选择“从零开始创建”后，单击 `创建` ，进入“创建流水线”页面。

4. 在“创建流水线”页面的画布中，根据本次业务需求配置“Source”和“Deploy”阶段后，依次在各阶段完成以下任务配置。



1. 在“Source”阶段添加项目代码源。

   在当前画布的“Source”卡片中，单击``添加代码源``，弹出“添加代码源”对话框。在该对话框中，配置代码源信息后，单击 保存 ，保存项目的代码源设置，并关闭对话框。



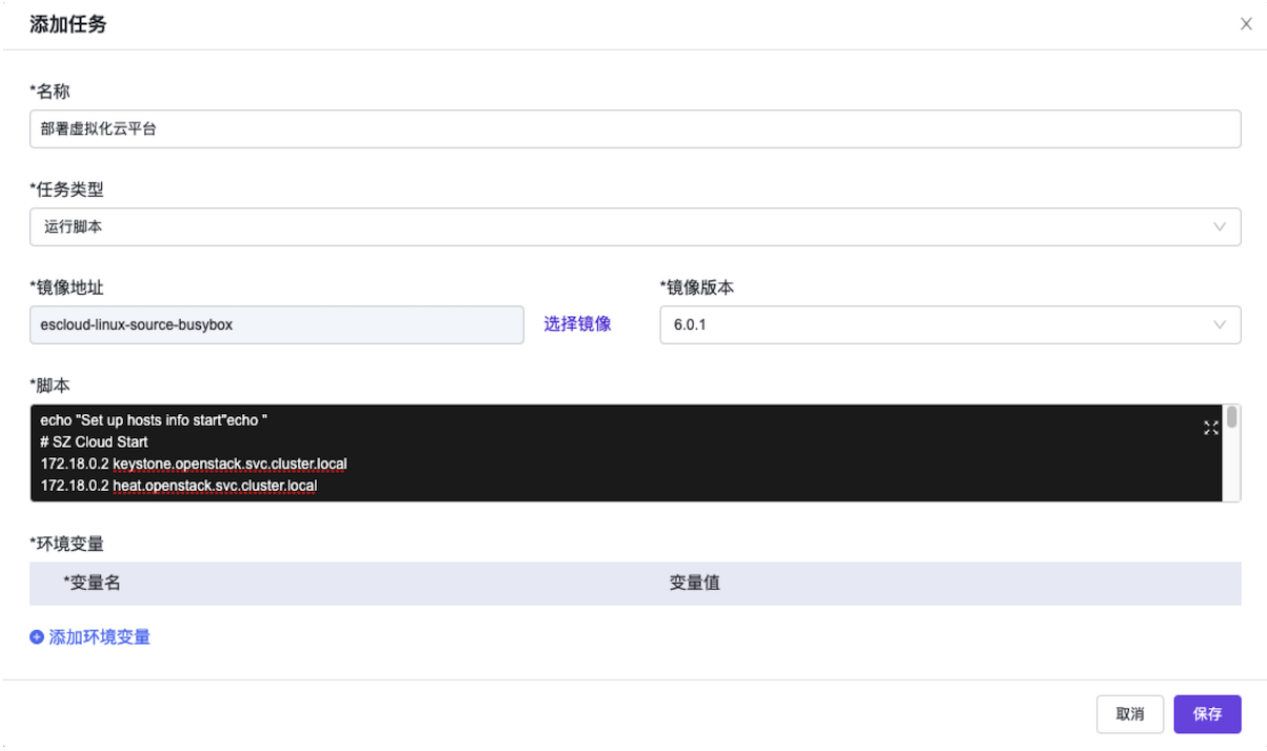
2. 在“Deploy”阶段添加“运行脚本”任务。

   在当前画布的“Deploy”卡片中，单击 `添加任务` ，弹出“添加任务”对话框。在该对话框中，“任务类型”选择“运行脚本”，“镜像地址”选择[自有镜像]页签中的“escloud-linux-source-busybox”，“镜像版本”选择“6.0.1”，并配置名称和脚本后，单击 `保存` ，完成任务创建，并关闭对话框。

   在上述“添加任务”对话框中，“脚本”输入内容如下。其中， *172.18.XX.XX* 为用于部署虚拟化云平台的云平台外部访问IP地址:

```
echo "Set up hosts info start"echo "
# SZ Cloud Start
172.18.XX.XX keystone.openstack.svc.cluster.local
172.18.XX.XX heat.openstack.svc.cluster.local
172.18.XX.XX neutron.openstack.svc.cluster.local
172.18.XX.XX nova.openstack.svc.cluster.local
172.18.XX.XX glance.openstack.svc.cluster.local
172.18.XX.XX cinder.openstack.svc.cluster.local
# SZ Cloud End
" >> /etc/hosts
echo "Set up hosts info done"

source ./openrc-sz
#get the correct version,through pkg server
VERSION=6.0.1
python update_version.py $VERSION
STACK_NAME=devops-deploy-$RANDOM-os-in-os
echo 'Try to create stack 3 nodes 3 cloud product nodes os in os env'
 heat stack-create $STACK_NAME -f ./stack-3nodes-3cloud-product-
nodes.yaml
# loop to check stack status
bash ./check_stack_status.sh  $STACK_NAME
```



5. 在“创建流水线”页面的画布中，单击画布右上方的 保存 后，在弹出的“保存”对话框中，选择保存方式后，单击 保存 ，完成流水线创建，并关闭当前页面。

3. 执行流水线。

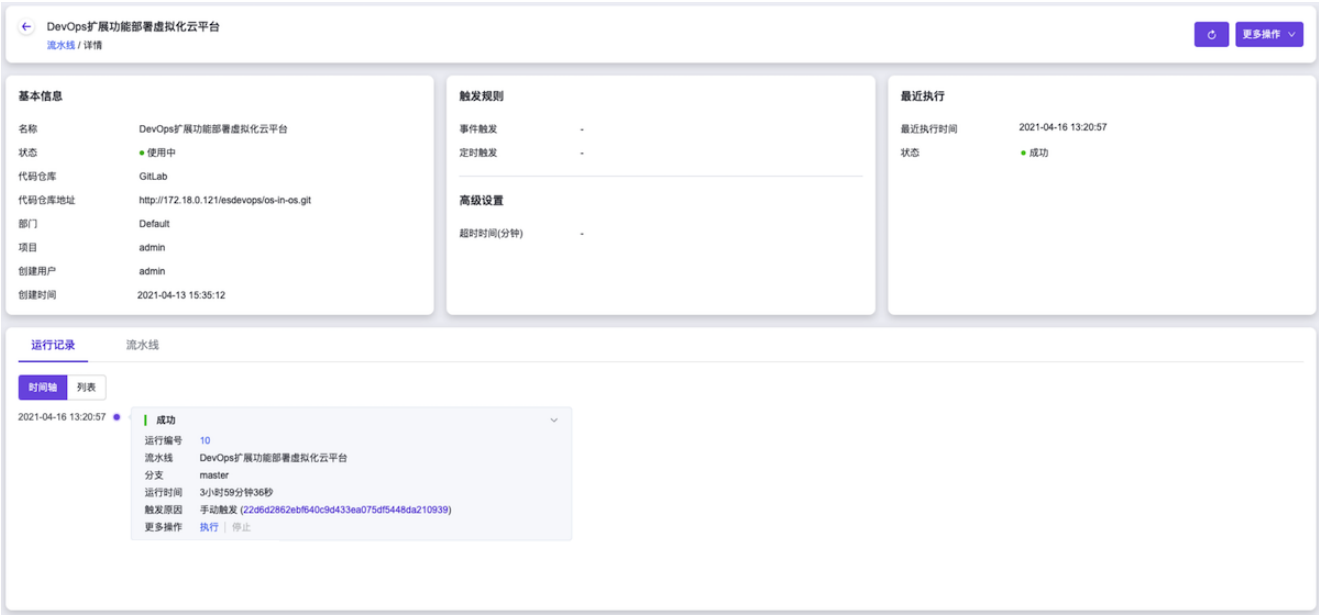
本实践方案中以手动触发方式为例，触发流水线执行。如需配置流水线自动触发，请参考 [配置流水线执行策略（可选）]。

1. 在“流水线”页面中，单击上述流水线所在行的 执行 ，弹出“执行流水线”提示框。
2. 在“执行流水线”提示框中，单击 执行 ，执行该流水线，并关闭提示框。

结果验证

1. 确认流水线成功执行。

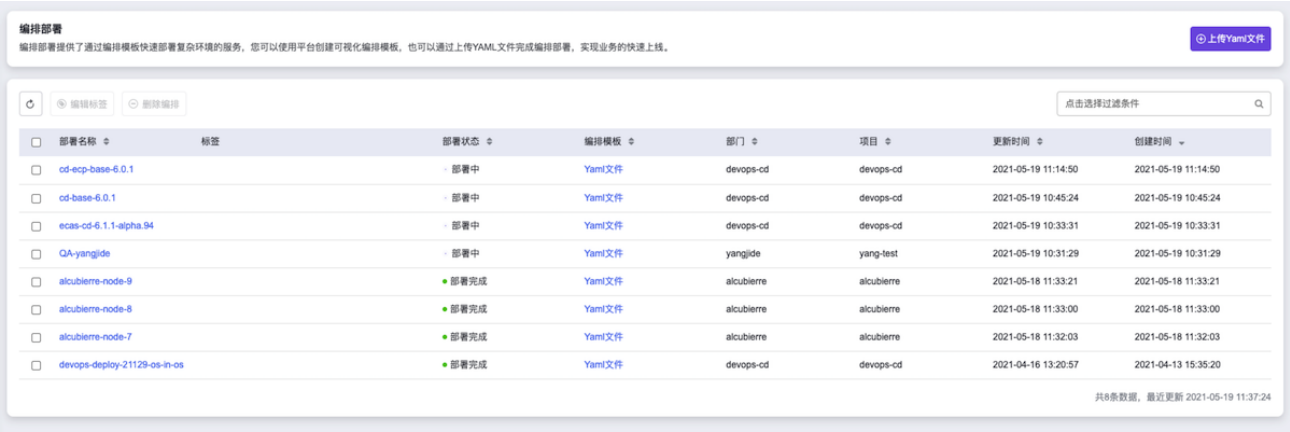
   在“流水线”页面中，单击上述流水线名称 ，进入其详情页面。在详情页面的[运行记录]页签中，确认该流水线执行成功。



2. 确认云平台成功编排部署。

   1. 在云平台的顶部导航栏中，依次选择[产品与服务]-[资源编排]-[编排部署]，进入“编排部署”页面。

   2. 在“编排部署”页面中，确认已生成名如 *devops-deploy-$RANDOM-os-in-os* 的部署，且部署状态为“部署完成”。



3. 确认云平台成功访问。

1. 在流水线执行完成后，会将执行结果以邮件形式发送至工程技术团队所有成员的个人邮箱中。在此邮件中，**Deployed stack name** 参数的值即为上述部署名称 *devops-deploy-$RANDOM-os-in-os*，**jumpserver fip** 参数的值即为虚拟化云平台的外网访问IP地址。

```
Job jenkins-X86-6.0.2-ECP-Install-71 result: SUCCESS
Please go to http://cicd.easystack.cn/job/X86-6.0.2-ECP-Install/71/ and verify the build.
Last stage:upload_images
Failed stage info:

##### Output from  pipeline #####
Deployed stack name: devops-deploy-21129-os-in-os
jumpserver fip: 172.18.[redacted]

devops-deploy-21129-os-in-os ( 172.18.[redacted] )
domain_name: ecp-install.example.io
##### Detailed Commits Info #####
```

2. 在本地计算机的浏览器地址栏中输入虚拟化云平台的外网访问IP地址，访问部署的虚拟化云平台。

咨询热线：400-100-3070

北京易捷思达科技发展有限公司：
北京市海淀区西北旺东路10号院东区23号楼华胜天成科研大楼一层东侧120-123
南京分公司：
江苏省南京市雨花台区软件大道168号润和创智中心B栋一楼西101
上海office：
上海黄浦区西藏中路336号华旭大厦22楼2204
成都分公司：
成都市高新区天府五街168号德必天府五街WE602

邮箱：
contact@easystack.cn (业务咨询)
partners@easystack.cn(合作伙伴咨询)
marketing@easystack.cn (市场合作)
training@easystack.cn (培训咨询)
hr@easystack.cn (招聘咨询)

版权所有© 北京易捷思达科技发展有限公司