

# 快速入门

## 情景

运维的职能为应用发布、变更、故障处理以及日常需求，覆盖研发运营生命周期的持续集成（CI）、持续部署（CD）、持续运营（CO）三个阶段，从测试包交付到测试环境，到将验证通过的版本部署到生产环境，以及提供业务上线后的运维基础和增值服务。

传统人工、脚本的运维模式会面临效率低下、频繁出错背锅等等挑战，在质量、效率、成本、安全四个维度上，无法跟随业务快速增长的步伐。此外，由于这种困境，运维交付基础运维服务已经疲于奔命，无法向上交付运维的增值服务，难以体现自身价值。

蓝鲸是腾讯游戏基于 PaaS 理念设计的一套研发运营一体化解决方案，服务于业务全生命周期。将服务以原子能力集成，并提供了自主研发 SaaS 的 DevOps 工具链，通过私有化部署的模式，帮助企业整合及落地研发运营体系，提升企业研发运营效率，帮助企业运维团队转型，提供更多增值服务。

接下来，我们通过一个传统单体应用的发布（将版本部署至生产环境）、故障处理（磁盘告警自动化处理）以及日常需求（测试同学自助调整应用配置）等几大场景，看蓝鲸是如何高效的交付运维服务。

## 前提条件

- 了解蓝鲸的体系架构

```
{% video %}http://bkopen-  
10032816.file.myqcloud.com/grayscale_test/BkVideo/bk_AdVideo_cn.mp4{% endvideo %}
```

- 完成蓝鲸的部署

## 操作步骤

- 准备环境
- 应用交付
- 故障处理
- 日常需求

## 准备环境

完成蓝鲸部署后，接下来开始初始化环境。

在蓝鲸 CMDB 中新建业务、划分业务拓扑，以及安装 Agent，完成蓝鲸对业务主机的纳管。

### CMDB 新建业务及划分业务拓扑

在蓝鲸中，蓝鲸 CMDB 位于原子平台层，通过蓝鲸 PaaS 的 ESB 模块，为上层 SaaS 提供统一的资源管理。

所以，针对业务或者主机等资源的管理之前，要先将其在 CMDB 中纳管。

首先，在配置平台中 [新建业务](#)，然后参照 [业务上线时 CMDB 如何管理主机](#) 完成业务拓扑的划分。

接下来，安装 Agent，蓝鲸的管控平台可以对主机实现包含文件分发、命令执行、数据上报的管控，同时主机会自动录入到蓝鲸 CMDB 对应的业务。

### 安装 Agent

参照 [管理直连网络区域的主机](#)，完成 Agent 安装。

### 执行脚本和分发文件

执行一个脚本和文件分发，验证环境是否已准备妥当。

```
{% video %}..//CD/Automation/media/blueking_execute_push_file.mp4{% endvideo %}
```

接下来，我们尝试做一次单体应用的版本发布。

## 应用交付

参照 [一次标准的应用交付自动化案例](#)，完成一次自动化的应用交付，效果如下：

```
{% video %}..//CD/Automation/media/sops_execution.mp4{% endvideo %}
```

完成应用交付后，接下来了解在蓝鲸中如何实现告警的自动化处理。

## 故障处理

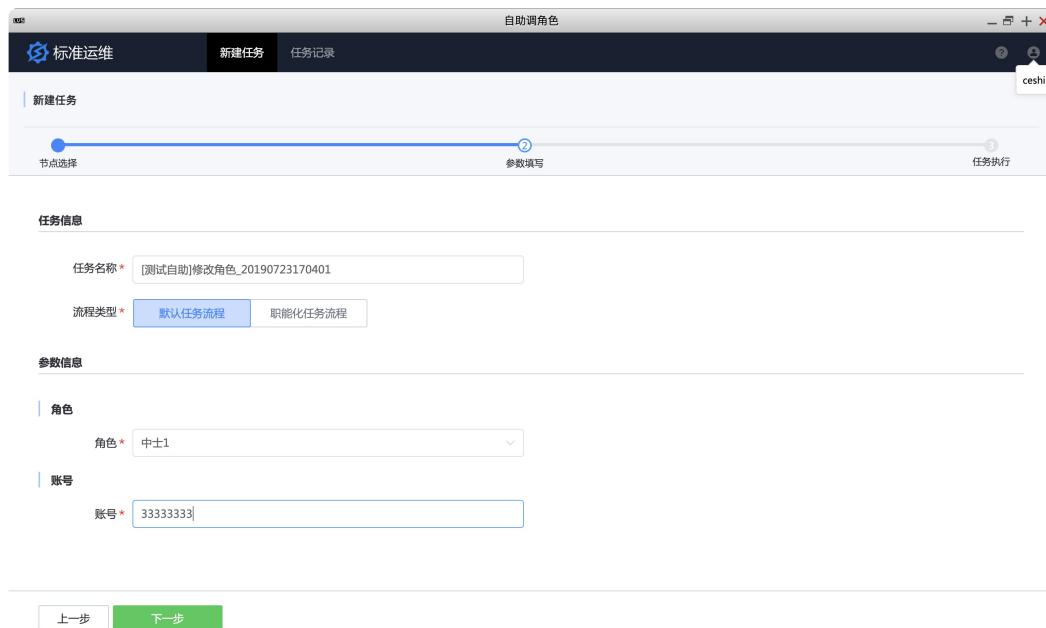
根据企业内部正在使用的监控系统，参照 [Zabbix 告警自动处理](#) 或 [第 3 方监控系统告警自动处理](#) 或 [蓝鲸监控告警自动处理](#)，实现 [磁盘告警自动化处理](#)。

```
{% video %}..//CO/FTA/media/zabbix_fta.mp4{% endvideo %}
```

接下来，了解来自产品、运营、测试等团队，对运维的日常需求，是如何释放运维，做到需求自助化的。

## 日常需求

参照 [需求自助化:测试自助调整验收环境](#)，完成测试自助修改测试环境配置，从此再也不会骚扰运维了。



以上就是一个运维的日常，在蓝鲸中如何做到 **标准化**、**自动化** 和 **自助化**。

当然，如果想做深，你还需要实现 [CMDB 中资源实例的自动发现](#)，靠自动化去维护 CMDB 实例。

在自动化的应用发布过程中，不免还是需要和上下游沟通版本发布细节，如此高的沟通成本，我们建议运维在标准运维固化发布流程，并 [启用操作员的发布功能](#)，让运维更关注增值服务。

另一方面，蓝鲸 ITSM 实现 [服务器资源申请流程线上化](#)、[应用发布流程线上化](#)、[环境变更流线上化](#)、[故障提报流程线上化](#)，通过线上流程完成企业 IT 的运营维护，告别邮件的低效率和不可回溯的交付模式。

从持续集成(Continuous Integration)、持续交付(Continuous Delivery)再到持续部署(Continuous Deployment)，蓝盾会逐渐完善覆盖，尽请期待

解决了运维当前的主要矛盾后，蓝鲸更关注 [研发运营这个领域的行业技术变化](#)。

通过支持 Kubernetes 和 Mesos 容器编排的蓝鲸容器管理平台，实现 [快速构建 Nginx 集群](#)、[应用的滚动升级](#)、[应用的蓝绿发布](#) 以及自动扩缩容等微服务化带来的技术红利。

蓝鲸，不止如此，更多尽情关注蓝鲸公众号。



## 测试环境自动更新

### 情景

开发同学完成功能开发、本地测试通过后，运维通过自动化工具或人工登录服务器更新，由于开发和运维存在 **沟通成本**，导致持续集成的测试环节效率低下，影响整体研发效率。

接下来看蓝盾（BK-CI）是如何做到 **测试环境自动更新**。

### 前提条件

- 部署蓝盾（BK-CI）

## 操作步骤

- 配置自动构建流水线
- 提交代码（Git Commit）验证

### 配置自动构建流水线



### 提交代码（Git Commit）验证

下面为一次自动更新记录，从 提交代码到更新测试环境，不到 1 分钟。

```
{% video %}assets/bk-ci-demo-HD.mp4{% endvideo %}
```

## 运维 "四化建设" 之标准化：配置管理标准化

“

自动化前，先做好标准化。

”

2014 年 GOT 大会上，腾讯游戏技术运营负责人刘栖铜在《腾讯游戏云的理想与实践》分享中提到运维服务的“四化建设”：标准化、自动化、服务化、产品化是如何交付运维价值。

这里我们介绍，运维“四化建设”的基石“标准化”中的 **配置管理标准化**，以 蓝鲸配置平台为案例，实战分析企业如何落地配置管理标准化，以及上层 **SaaS** 如何消费 **CMDB**，为后续的自动化、数据化、智能化打牢基础。

首先，我们看下 IT 基础资源、应用在配置管理方面遇到的痛点。

## 配置管理的痛点

- 手工录入 CMDB 数据
- 日常的发布、变更、故障处理、日常需求处理等运维自动化系统没有消费 CMDB，而是在

CMDB 和自动化系统中通过键盘复制粘贴主机 IP...

- 无法保证配置数据的准确性
- 未纳管应用系统，仅管理 IT 基础设施，如主机、网络设备等
- 排查问题时，无法通过可视化拓扑方式查看资源间的关联关系，比如 DB 挂了，影响了哪些业务的哪些模块
- ...

接下来，我们尝试以项目的方式，解决运维在配置管理上遇到的痛点。

## 项目目标

建立一个权威统一的 CMDB 配置主数据，作为自动化运维平台、监控平台、运维流程平台等一系列 IT 系统的基石。

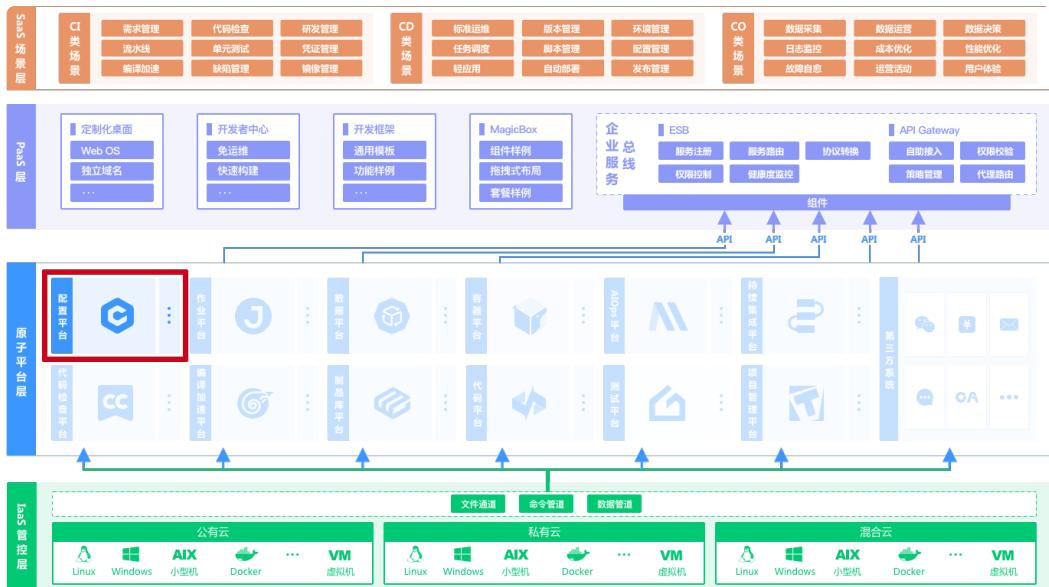


CMDB 是 ITIL 的基石

确立了项目目标，接下来做技术选型。

## 技术选型

在蓝鲸体系架构中，配置平台位于原子平台层，通过蓝鲸 PaaS 的 ESB 为上层 SaaS 提供覆盖研发运营 CI（持续集成）、CD（持续交付和持续部署）、CO（持续运营）领域的配置管理能力。



配置平台 在蓝鲸体系架构中的位置

这意味着，在架构上确立了蓝鲸配置平台可打破企业研发运营系统存在的“信息孤岛，无法实现配置管理数据互通”的痛点，不给企业再新增一个竖井式的“烟囱”。

以下是蓝鲸配置平台（[已开源](#)）相当于传统 CMDB 的优势，正好切中运维在配置管理上遇到的痛点，同时在产品设计和架构上具有很强的扩展能力。



配置平台 的核心能力

确立了技术方案后，接下来介绍如何在企业内部实施 CMDB，使其成为自动化等 IT 系统的基石。

如何实施 CMDB，请参考场景案例：[业务上线时 CMDB 如何管理主机](#)、[CMDB 如何管理进程](#)、[CMDB 如何管理 MySQL 实例](#)、[企业 CMDB 主机实例同步至蓝鲸 CMDB](#)、[自动发现 MySQL 实例](#)。

## 技术发展方向

当自动化等 IT 系统越来越依赖于 CMDB，如何保证 **CMDB 数据的准确性** 呢？我们认为 CMDB 需要具备配置数据的审计和校验能力，并关联 ITSM 系统，完善 CMDB 中配置数据的变更流程。

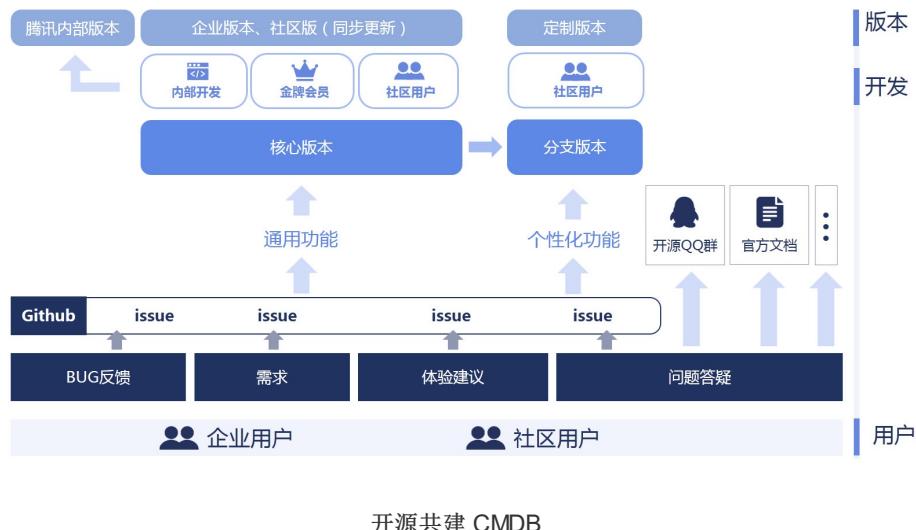
自动发现采集器的开发毕竟存在开发成本，是否可以提供 **开箱即用的 CI 属性和 CI 间关联关系**（如数据库或中间件运行在哪台主机上，亦或是业务程序访问了哪台 DB）的自动发现能力。

此外，IT 系统对 **CMDB 的消费程度需要持续加深**，CMDB 需加强在 CI 属性、关联关系以及供上层 SaaS 消费的便利度，比如监控系统中的资源对象全部以 CMDB 中为准，不再自己保留一部分资源对象，真正实现以 CMDB 为核心的资源管理和自动化运维。

万丈高楼平地起，无论想实现自动化、数据化还是智能化，都需要把第一步的标准化做好，尤其是标准化中的配置管理标准化，切不可操之过急。

最后希望，企业的 IT 系统建设能够逐步演进、完善，让广大运维兄弟真正能 **交付运维价值**，体会运维这个岗位给企业带来的价值。

来吧，一起共建属于大家的 **开源的 CMDB 项目**！



## 扩展阅读

- 论运维的四化建设 腾讯游戏的实践分享
- Configuration Management Database
- Github Tencent/bk-cmdb

# 业务上线时 CMDB 如何管理主机

## 情景

新业务（常见的三层架构：接入层、逻辑层、存储层）上线，需要用 CMDB 管理业务上线依赖的主机资源，便于后续实现发布、变更、故障处理等场景的自动化流程。

## 前提条件

在配置平台中 [新建完业务](#)，并导入主机及将主机分配到业务中。

## 操作步骤

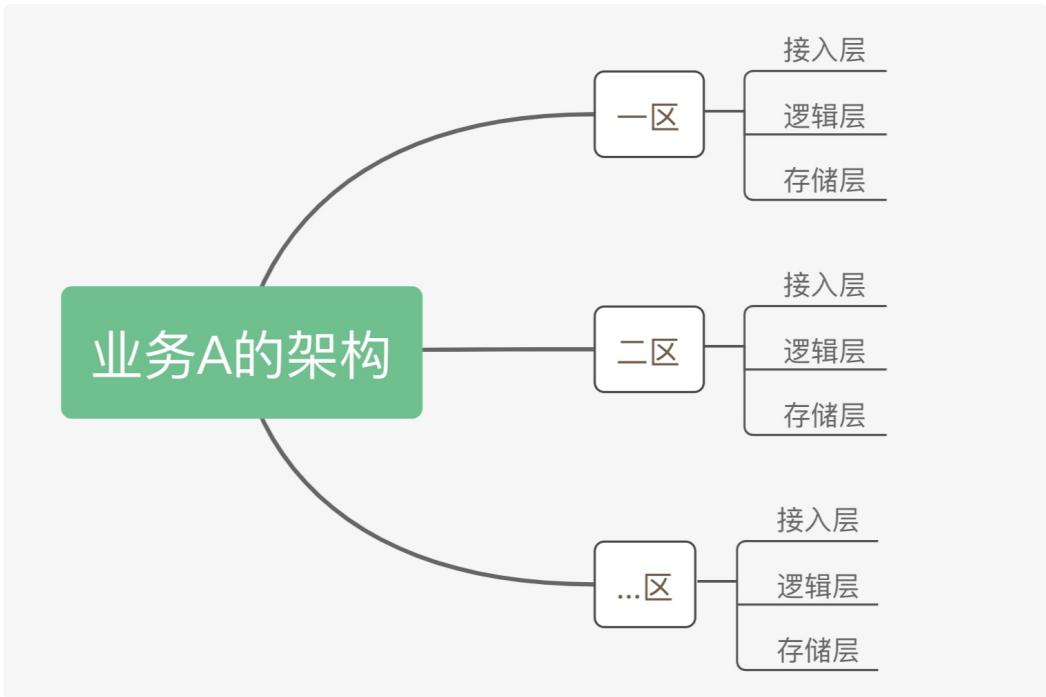
- 梳理业务架构
- 新建集群
- 新建模块
- 使用场景：在作业平台中查询接入层的磁盘使用率

### 梳理业务架构

业务的架构设计最终要体现在 CMDB 的业务拓扑树上，所以我们先梳理业务的架构。

以业务“欢乐游戏( demo )"为例，其架构是常见的三层架构：[接入层](#) -> [逻辑层](#) -> [存储层](#)

为了满足全国各地用户就近接入，按照地域划分了不同的区，但架构一致。

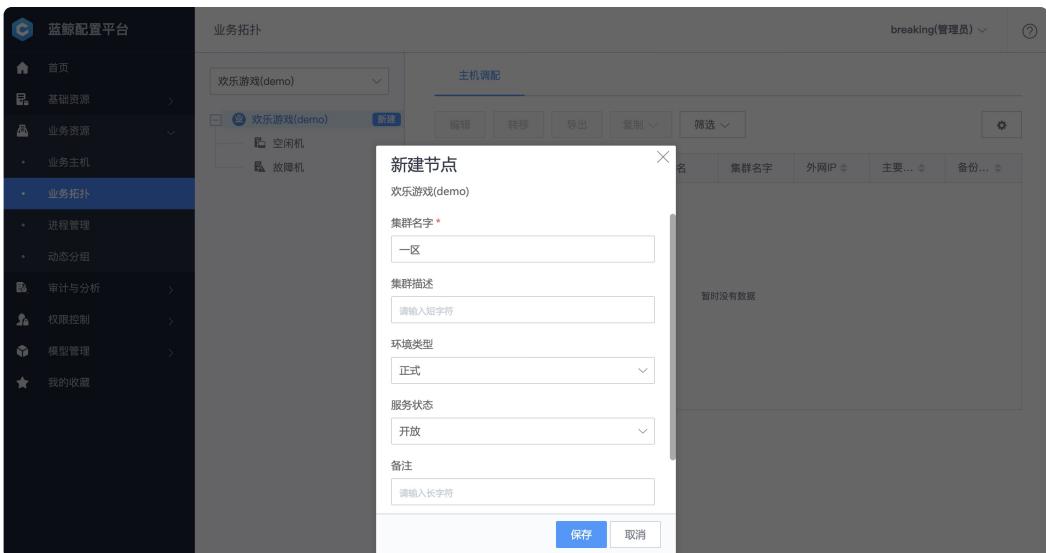


## 新建集群

按照上述业务架构，可将做如下对应：

- 业务中的区 <-> CMDB 中的集群
- 业务中的分层架构 <-> CMDB 中模块

在 **业务资源 -> 业务拓扑** 中，选中根节点 **业务**，新建节点 **集群** 一区。



“

环境类型：一般保持 **正式**；测试环境设置为 **测试**，让测试环境的发布流程模板只能选

择 **测试** 集群；

服务状态：一般保持 **开放**，在某些场景如开区准备的时候，状态可置为 **关闭**，让发布系统无法选中；

”

新建完集群 **一区**、**二区** 后，可看到如下业务拓扑

接下来，在集群下创建模块。

## 新建模块

选中 **集群** 节点，新建模块

“

模块类型：一般为 **普通**，当模块类型为数据库类时，选择 **数据库** 主要维护人、备份维护人：该模块维护人，一般在告警推送时会用到；区别在于电话通知时优先 **主要维护人**，如果 **主要维护人** 未接听，自动转 **备份维护人**，此处逻辑需要自行实现；

”

模块创建好了，接下来 **分配主机**，将 **基础资源** -> **主机** 中的资源按照业务架构分配至对应的模块；

## 使用场景：在作业平台中查询接入层的磁盘使用率

通过一个简单的场景，体验 **作业平台** 如何消费主机实例 { % video % } media/cmdb\_job\_consume.mp4 { % endvideo % }

## 扩展阅读

### 蓝鲸内置 SaaS 的 CMDB 消费场景

#### 应用发布、变更：资源编排工具 标准运维

应用发布、变更流程包含版本在多台主机上的文件分发、命令执行等操作，如何优雅的选择这批主机，需要使用 CMDB 的查询主机实例功能。



(在标准运维中一次应用交付的执行历史)

在参数中使用 **动态IP** 变量，运维无需关心主机扩缩容、故障替换等场景带来的主机变更，无需担心漏更新主机或更新错主机。

## 参数信息

版本发布目标IP (i)

版本发布目标IP \*  静态IP  动态IP

已选择 2 个节点

×

搜索节点

×

- 欢乐游戏(demo)
  - 空闲机池
    - 空闲机
    - 故障机
  - Android\_Weixin
    - 一区
      - 接入层
      - 逻辑层
      - 存储层
    - 二区
      - 接入层
      - 逻辑层
      - 存储层
  - Android\_QQ

筛选条件 ( 同时满足 ) :

增加一条筛选条件

排除条件 ( 同时满足 ) :

集群

二区

+

-

上一步

下一步

## 故障处理：监控、故障自愈

针对业务架构中的某一层级模块（如接入层）设置一个 **告警检测策略**，无需关心实例的新增、删除及修改。

## 性能指标

\* 指标分类

磁盘

指标名称

磁盘使用率

采集周期: 1分钟

## 触发条件

\* 监控对象  按主机IP  按业务拓扑



如何实现实时感知，背后的逻辑是通过 CMDB 的事件推送功能，实时感知实例的新增、修改、删除等动作。

The screenshot shows two main parts of a software interface. On the left is a table titled '事件推送' (Event Push) with columns: 推送名称 (Push Name), 系统名称 (System Name), 操作人 (Operator), and 更新时间 (Update Time). It lists two entries: 'gse\_agentid' (operator: migrate, time: 2019-05-11 02:35) and 'process instance refresh [Do not remov...' (operator: cmd0, time: 2019-05-11 02:35). Below the table are pagination controls: 总 2 行 第 1 / 1 页 每页显示 10 行. On the right is a modal window titled '编辑推送' (Edit Push) with fields: 推送名称 (Push Name: gse\_agentid), 系统名称 (System Name: gse), URL (URL: http://10.0.4.51:52050/gsesynodata/v1/c), and a '测试推送' (Test Push) button. Below these are sections for '成功确认方式' (Success Confirmation Method) with radio buttons for 'HTTP状态' (HTTP Status) and '正则验证' (Regular Expression Validation), and a text input for 'HTTP状态' (HTTP Status: 200). There are also sections for '主机业务' (Host Business), '业务拓扑' (Business Topology), and '组织架构' (Organizational Structure), each with checkboxes for various actions like '全选' (Select All), '新增' (Add), '编辑' (Edit), and '删除' (Delete).

在 **故障自愈** 中的消费场景也是如此，一个或多个模块的某一个告警，关联对应的处理动作。

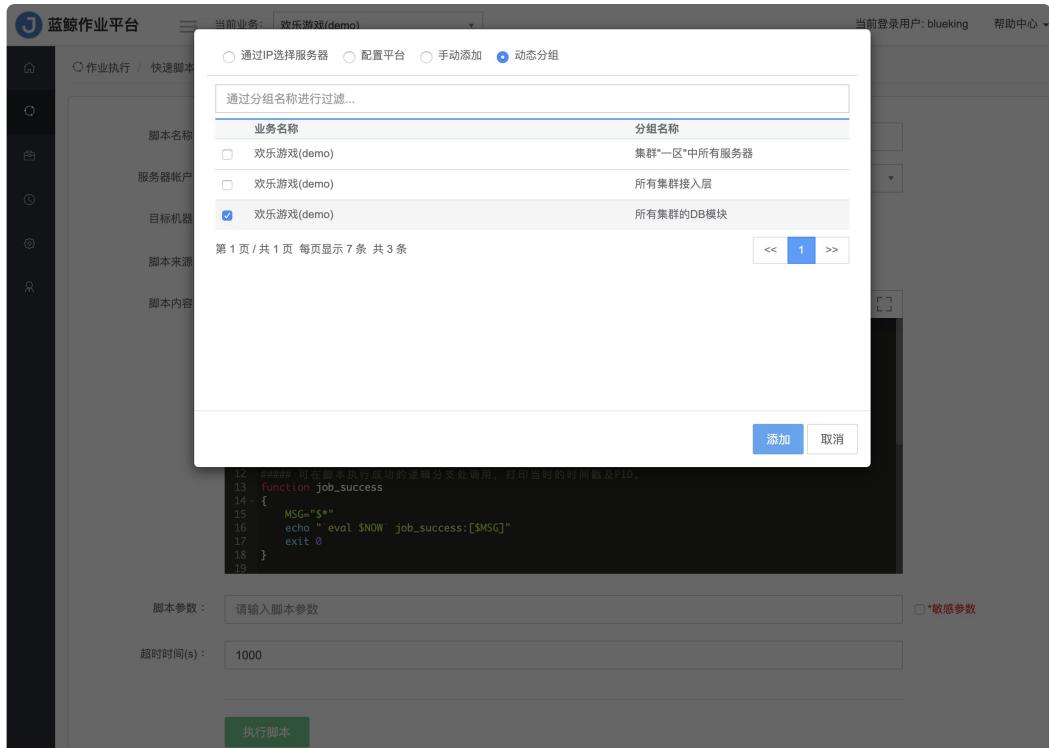
The screenshot shows the '故障自愈' (Fault Recovery) interface. At the top is a navigation bar with a shield icon, the text '故障自愈', and a dropdown menu '欢乐游戏(demo)'. Below the navigation is a sidebar with icons for home, dashboard, topology, logs, metrics, and users. The main content area has two sections: '自愈场景' (Recovery Scenario) and '自愈处理' (Recovery Action). In '自愈场景', there are filters for '告警类型' (Alarm Type: [主机监控] 磁盘使用率), '平台' (Platform: 默认全选), and '模块' (Module: 接入层). In '自愈处理', there is a dropdown for '自愈套餐' (Recovery Package: 接入层磁盘清理) with a '查看' (View) button and a '+' button to add more packages.

## 平台团队对资源的管控，例如全业务 DBA、AIX SA

以 DBA 为例，需要管控所有 DB，可将运行数据库的主机所属模块的类型设置为 **数据库**，然后通过 **动态分组** 功能查询模块类型为 **数据库** 的主机。

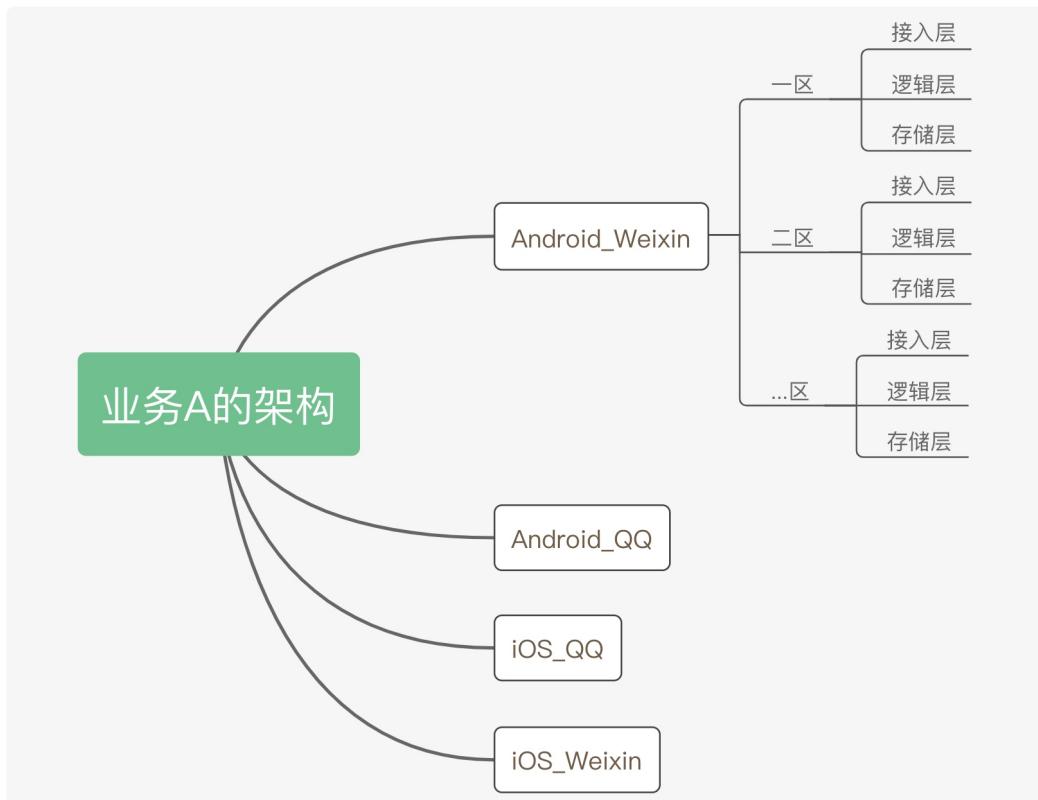
The screenshot shows the '蓝鲸配置平台' (Blue Whale Configuration Platform) with a dark theme. The left sidebar includes '首页' (Home), '基础资源' (Basic Resources), '业务资源' (Business Resources) with sub-options '业务主机', '业务拓扑', and '进程管理', and '动态分组' (Dynamic Grouping) which is selected. The main content area shows a '动态分组' (Dynamic Grouping) table with columns: ID, 分组名称 (Group Name), 创建用户 (Create User), and 创建时间 (Create Time). It lists three groups: '集群“一区”中所有服务器' (All servers in cluster 'One Zone'), '所有集群接入层' (All cluster access layers), and '所有集群' (All clusters). Below the table is a '预览分组' (Preview Group) table with columns: 内网IP (Internal IP), 集群 (Cluster), 模块 (Module), 业务 (Business), and 云区域 (Cloud Region). It shows one entry: '10.0.4.29' (内网IP), '一区' (Cluster), '存储层' (Storage Layer), '欢乐游戏(demo)' (Business), and 'default area' (Cloud Region). A modal window titled '编辑分组' (Edit Group) is open on the right, showing fields for '业务' (Business: 欢乐游戏(demo)), '分组名称' (Group Name: 所有集群的DB模块), and '分组内容' (Group Content: 匹配IP,集群,模块,业务,云区域). It also shows a dropdown for '模块类型' (Module Type: 于) with '数据库' selected. Buttons at the bottom of the modal include '查询' (Search), '保存' (Save), '取消' (Cancel), and '删除' (Delete).

在 **蓝鲸作业平台** 中可使用该动态分组来选择 DB 主机。

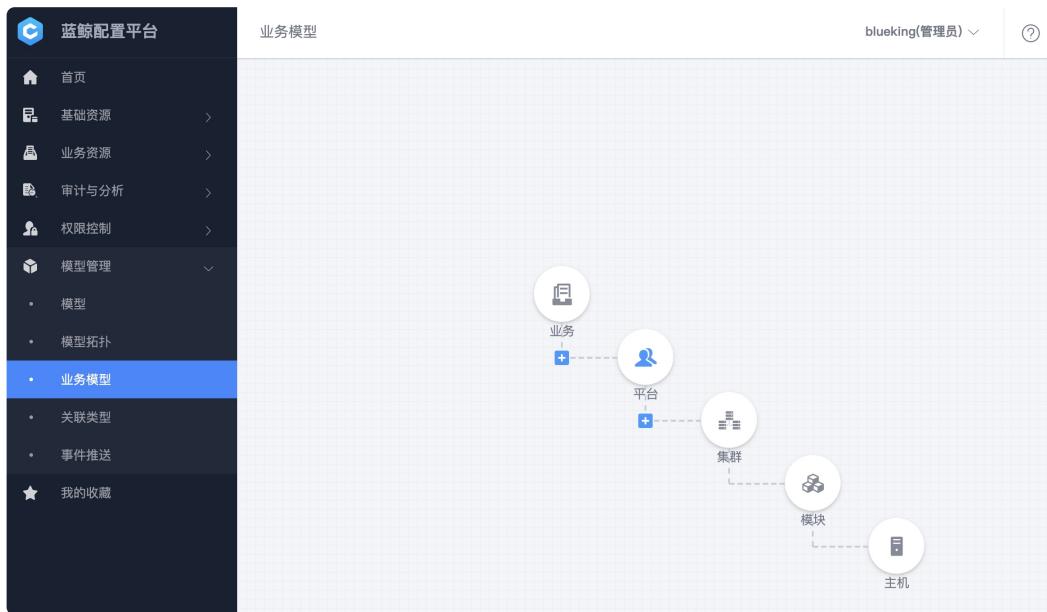


### 三级拓扑不够用，怎么办？新建多级拓扑节点

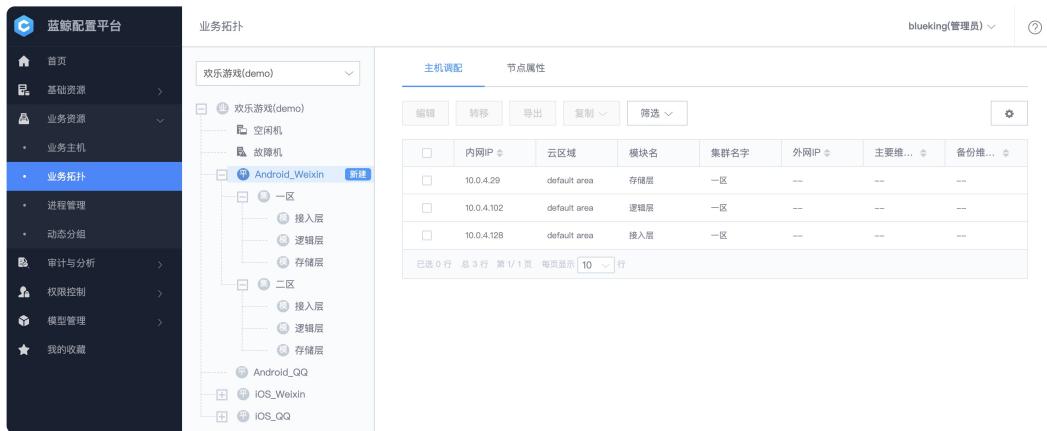
如果业务架构在大区（集群）之上还有一级（平台：如 `Android_Weixin`、`Android_QQ` 等）



可在 **业务模型** 中，**业务** 与 **集群** 间新建一级或多级拓扑



新的业务拓扑如下：



“注：当前版本新增层级拓扑的生效范围是所有业务，在未来的某一个版本中可调整生效范围；蓝鲸体系在未来的某一个迭代中，将重点弱化业务和细化权限”

## Master DB 和 Slave DB 如何实现物理架构的高可用？

两者一定不要放在同一个机架或交换机下，否则机架或交换机掉电，亦或是交换机故障，Master DB 和 Slave DB 都宕机，无法正常切换，导致业务无法对外提供服务。

操作方法：[模型管理] -> [模型] -> [主机] -> [模型字段]，新增 **存放机架 ID**、**网络设备 ID**。

此外，配置主机与机架或上联交换机的关联可使用 **模型关联**，确保数据的唯一性可设置 **唯一校验**，另外 **字段分组** 可调整 CI 属性的呈现方式。

The screenshot shows the 'Business Host' configuration page in the Configuration Platform. On the left, there's a sidebar with navigation links like '首页', '基础资源', '业务资源', '业务主机' (which is selected), '业务拓扑', '进程管理', '动态分组', '审计与分析', '权限控制', '模型管理', and '我的收藏'. The main content area has tabs for '编辑', '转移', '导出', and '复制'. Below these tabs is a table with columns: 内网IP, 云区域, 模块名, and 集群名字. There are four rows in the table with values: 10.0.4.29, default area, 存储层, 一区; 10.0.4.102, default area, 贮存层, 一区; 10.0.4.128, default area, 接入层, 一区. At the bottom of the table, it says '已选 0 行 总 3 行 第 1 / 1 页 每页显示 10 行'. To the right of the table, there's a detailed view for host '10.0.4.29'. It includes sections for '属性' (Properties), '关联' (Associations), '实时状态' (Real-time Status), and '变更记录' (Change Log). The '属性' section contains sections for '业务拓扑' (Business Topology) and '基础信息' (Basic Information). Under '基础信息', fields include: 内网IP: 10.0.4.29, 外网IP: —, 团队编号: —, 主要维护人: —, 业务模块类型: 虚拟机, 备份维护人: —, 备注: —, 设备SN: —, SLA级别: —, 售保年限: —, 所属区域: default area, 所在国家: —, 所属运营商: —, 存放机架ID: 2003, 网络设备ID: 1003. A red box highlights the '存放机架ID: 2003' field. Below this is a section for '自动发现信息 (需要安装agent)' (Automatic discovery information (agent required)). It lists system details: 主机名称: storage\_layer\_breaking, 操作系统名称: linux centos, 操作系统版本: 7.5.1804, 操作系统位数: 64-bit, CPU频率: 1999 Hz, 内存容量: 991 MB, 内网MAC地址: 52:54:00:ec:c4:6e, 操作系统类型: Linux, CPU型号: AMD EPYC Processor, 磁盘容量: 49 GB, 外网MAC: —. A blue button labeled '编辑' is at the bottom.

## CMDB 如何管理进程

### 情景

应用的存储是 MariaDB，在 CMDB 中注册 MariaDB，以便在监控系统做进程监控。

### 前提条件

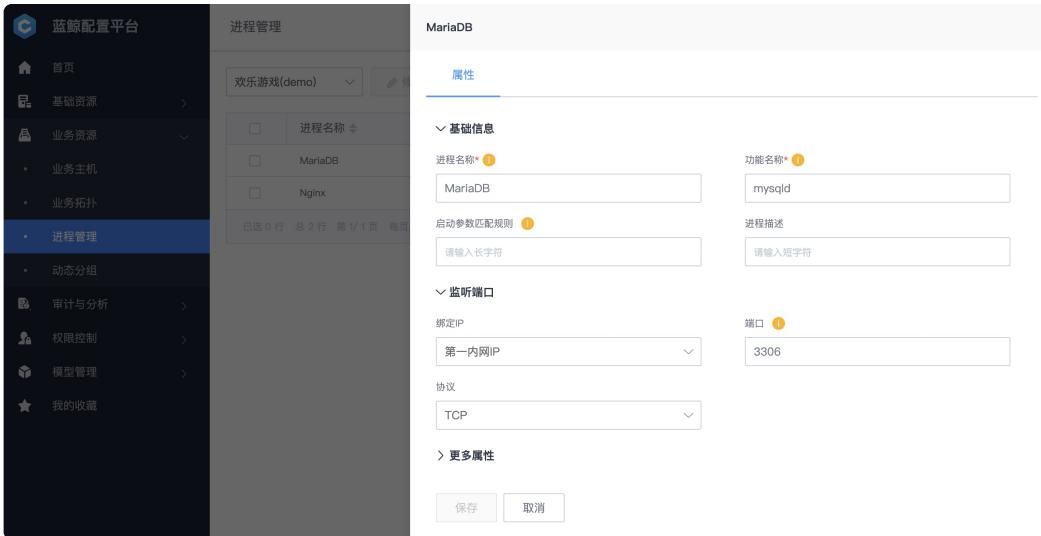
在配置平台中**新建业务** 及业务拓扑。

### 步骤

- CMDB 中注册进程和端口
- 将进程绑定模块
- 监控系统自动实现进程端口监控

### CMDB 中注册进程

在菜单 **业务资源 - 进程管理** 中 **新建进程**



## 注册进程

- 进程名称: 对外显示的服务名: `MariaDB`;
- 功能名称: 程序的二进制名称: `mysqld`;

在配置功能名称时, 使用命令查询该程序的二进制名称

```
$ ps -ef | grep -i mysqld
mysql      7800      1  0 7月08 ?        00:00:00 /bin/sh /usr/bin/mysqld_safe --basedir=/
usr
mysql      7980  7800  0 7月08 ?        00:01:55 /usr/libexec/mysqld --basedir=/usr --dat
adir=/var/lib/mysql --plugin-dir=/usr/lib64/mysql/plugin --log-error=/var/log/mariadb/m
ariadb.log --pid-file=/var/run/mariadb/mariadb.pid --socket=/var/lib/mysql/mysql.sock
```

注: 监控系统一般完全匹配二进制名称。获取二进制名称的方法: `basename $(readlink -f /proc/7980/exe)`

## 注册端口

查询 mysqld 监听的 IP 和端口

```
$ netstat -antp | grep mysqld
tcp        0      0 10.0.4.29:3306          0.0.0.0:*          LISTEN      7980/my
sqld
```

- 绑定 IP: MariaDB 为存储层, 一般绑定内网 IP, 故选择 `第一内网 IP`。
- 端口: 进程监听的端口, `3306`
- 协议: `TCP`

## 将进程绑定模块

如何管理主机 提到依据业务架构来划分业务拓扑，业务拓扑中模块代表服务，而服务将由一个或多个进程监听的端口来对用户或其他模块提供服务。

所以，需要将 **进程** 绑定至对应的 **模块** 上。

这里将 **MariaDB** 进程绑定至存储层模块上。

The screenshot shows the 'Process Management' interface. On the left, under '业务拓扑' (Business Topology), there is a tree view of the system architecture. In the center, the 'MariaDB' process is selected. On the right, the 'Module Binding' tab is active, showing a table with three rows: '存储层' (Storage Layer) is bound to the '逻辑层' (Logical Layer) and has a status of '已绑定' (Bound); '接入层' (Access Layer) and '逻辑层' (Logical Layer) are unbound ('未绑定').

## 监控系统自动实现进程端口监控

给模块 **分配主机**

The screenshot shows the 'BlueKing Configuration Platform'. The left sidebar is the navigation menu. In the center, under '业务拓扑' (Business Topology), a tree view shows the system architecture. On the right, the 'Host Allocation' section is displayed, showing a table with one row: '10.0.4.29' is assigned to the 'default area' in the '一区' (Area 1) and is associated with the '存储层' (Storage Layer) module.

等候一分钟，在蓝鲸自带的监控系统 **蓝鲸监控** 中可以看到 **进程的运行情况**。

The screenshot shows the 'BlueKing Monitoring System'. At the top, there are various filtering and alerting options. The main table displays three hosts: '10.0.4.102' (告警状态: 数据未上报, CPU使用率: 0%, 磁盘I/O使用率: 无查询数据, CPU5分钟负载: 0.01, 进程服务: System, MariaDB), '10.0.4.29' (告警状态: 正常, CPU使用率: 1.67%, 磁盘I/O使用率: 0.12%, CPU5分钟负载: 0.01, 进程服务: System, MariaDB), and '10.0.4.128' (告警状态: 正常, CPU使用率: 1.40%, 磁盘I/O使用率: 0.49%, CPU5分钟负载: 0.01, 进程服务: System, Nginx). The 'MariaDB' service icon is highlighted with a red box.

点击 **MariaDB** 图标，可以查看其占用的 CPU、内存使用率以及文件句柄数等进程占用的资源指标。



## 扩展阅读

### 监控系统消费 CMDB 中进程配置背后的逻辑

当给模块分配完主机后，该主机的大部分 CI 属性将被自动推送至 `/var/lib/gse/host/hostid`

```
{
    .....
    "process" : [
        {
            "bind_ip" : "3",
            "bind_modules" : [ 68 ],
            "bk_func_id" : "",
            "bk_func_name" : "mysqld", // 对应功能名称
            "bk_process_id" : 110,
            "bk_process_name" : "MariaDB", // 对应进程名称
            "bk_start_param_regex" : "",
            "port" : "3306",
            "protocol" : "1"
        }
    ]
}
```

蓝鲸内置的进程监控采集器 `processbeat` 会读取该文件，并写入自身的配置文件 `/usr/local/gse/plugins/etc/processbeat.conf`，以实现进程和端口的监控。

```
.....
processbeat.processes:
- name: mysqld // 二进制名称
  displayname: MariaDB
  protocol: tcp
  ports:
  - 3306
```

```
paramregex: ""
bindip: 10.0.4.29
```

## 二进制名称均为 java，该如何配置

如 ZooKeeper、Hadoop 的二进制均为 `java`，可用进程启动参数区分

```
$ ps -ef | grep -i zookeeper

root      5897      1  0 Nov14 ?          01:49:00 /data/bkce/service/java/bin/java -Dzook
eepер.log.dir=/data/bkce/logs/zk/ -Dzookeeper.root.logger=INFO,ROLLINGFILE -Dzookeeper.
DigestAuthenticationProvider.superDigest=bkadmin:1bf5dHUwvnyrhMDaPlkHwFS1J0g= -cp /data
/bkce/service/zk/bin/../build/classes:/data/bkce/service/zk/bin/../build/lib/*.jar:/dat
a/bkce/service/zk/bin/../lib/slf4j-log4j12-1.6.1.jar:/data/bkce/service/zk/bin/../lib/s
lf4j-api-1.6.1.jar:/data/bkce/service/zk/bin/../lib/netty-3.10.5.Final.jar:/data/bkce/s
ervice/zk/bin/../lib/log4j-1.2.16.jar:/data/bkce/service/zk/bin/../lib/jline-0.9.94.jar
:/data/bkce/service/zk/bin/../zookeeper-3.4.10.jar:/data/bkce/service/zk/bin/../src/jav
a/lib/*.jar:/data/bkce/etc:/data/bkce/service/zk/conf:/data/bkce/service/java/lib: -Dco
m.sun.management.jmxremote -Dcom.sun.management.jmxremote.local.only=false org.apache.z
ookeeper.server.quorum.QuorumPeerMain /data/bkce/etc/zoo.cfg

root      10220      1  2 Nov14 ?          13:17:41 /data/bkce/service/java/bin/java -Xmx1G
-Xms1G -server -XX:+UseG1GC -XX:MaxGCPauseMillis=20 -XX:InitiatingHeapOccupancyPercent=
35 -XX:+DisableExplicitGC -Djava.awt.headless=true -Xloggc:/data/bkce/logs/kafka/kafkaS
erver-gc.log -verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCTimeSta
mps -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.authenticate=false -D
com.sun.management.jmxremote.ssl=false -Dkafka.logs.dir=/data/bkce/logs/kafka -Dlog4j.c
onfiguration=file:../config/log4j.properties -cp /data/bkce/service/java/lib:/data/b
kce/service/kafka/bin/../libs/aopalliance-repackaged-2.5.0-b05.jar:/data/bkce/service/k
afka/bin/../libs/argparse4j-0.7.0.jar:/data/bkce/service/kafka/bin/../libs/connect-api-
0.10.2.0.jar:/data/bkce/service/kafka/bin/../libs/connect-file-0.10.2.0.jar:/data/bkce/
service/kafka/bin/../libs/connect-json-0.10.2.0.jar:/data/bkce/service/kafka/bin/../lib
s/connect-runtime-0.10.2.0.jar:/data/bkce/service/kafka/bin/../libs/connect-transforms-
0.10.2.0.jar:/data/bkce/service/kafka/bin/../libs/guava-18.0.jar:/data/bkce/service/kaf
ka/bin/../libs/hk2-api-2.5.0-b05.jar:/data/bkce/service/kafka/bin/../libs/hk2-locator-2
.5.0-b05.jar:/data/bkce/service/kafka/bin/../libs/hk2-utils-2.5.0-b05.jar:/data/bkce/se
rvice/kafka/bin/../libs/jackson-annotations-2.8.0.jar:/data/bkce/service/kafka/bin/..l
ibs/jackson-annotations-2.8.5.jar:/data/bkce/service/kafka/bin/../libs/jackson-core-2.8
.5.jar:/data/bkce/service/kafka/bin/../libs/jackson-databind-2.8.5.jar:/data/bkce/servi
ce/kafka/bin/../libs/jackson-jaxrs-base-2.8.5.jar:/data/bkce/service/kafka/bin/..l
ibs/jackson-jaxrs-json-provider-2.8.5.jar:/data/bkce/service/kafka/bin/..l
ibs/jackson-module-jaxb-annotations-2.8.5.jar:/data/bkce/service/kafka/bin/..l
ibs/javassist-3.20.0-GA.jar:/data/bkce/service/kafka/bin/..l
ibs/javax.annotation-api-1.2.jar:/data/bkce/service/kafka/bin/..l
ibs/javax.inject-1.jar:/data/bkce/service/kafka/bin/..l
ibs/javax.inject-2.5.0-b05.jar:/data/bkce/service/kafka/bin/..l
ibs/javax.servlet-api-3.1.0.jar:/data/bkce/service/kafka/bin/..l
ibs/javax.ws.rs-api-2.0.1.jar:/data/bkce/service/kafka/bin/..l
ibs/jersey-client-2.24.jar:/data/bkce/service/kafka/bin/..l
ibs/jersey-common-2.24.jar:/data/bkce/service/kafka/bin/..l
ibs/jersey-container-servlet-2.24.jar:/data/bkce/service/kafka
/bin/..l
ibs/jersey-container-servlet-core-2.24.jar:/data/bkce/service/kafka
/bin/..l
ibs/jersey-guava-2.24.jar:/data/bkce/service/kafka/bin/..l
ibs/jersey-media-j
xb-2.24.jar:/data/bkce/service/kafka/bin/..l
ibs/jersey-server-2.24.jar:/data/bkce/service/kafka
/bin/..l
ibs/jetty-continuation-9.2.15.v20160210.jar:/data/bkce/service/kafka
/bin/..l
ibs/jetty-http-9.2.15.v20160210.jar:/data/bkce/service/kafka
/bin/..l
ibs/jetty-io-9.2.15.v20160210.jar:/data/bkce/service/kafka
/bin/..l
ibs/jetty-security-9.2.15.v20160210.jar:/data/bkce/service/kafka
/bin/..l
ibs/jetty-server-9.2.15.v20160210.jar:/data/bkce/service/kafka
/bin/..l
ibs/jetty-servlet-9.2.15.v20160210.jar:/data/bkce/service/kafka
/bin/..l
ibs/jetty-servlets-9.2.15.v20160210.jar:/data/bkce/service/kafka
/bin/..l
ibs/jopt-simple-5.0.3.jar:/data/bkce/service/kafka
/bin/..l
ibs/kafka_2.12-0.10.2.0-sources.jar:/data/bkce/service/kafka
/bin/..l
ibs/kafka-clients-0.10.2.0.jar:/data/bkce/service/kafka
/bin/..l
ibs/kafka-log4j-appender-0.10.2.0.jar:/data/bkce/service/kafka
/bin/..l
ibs/kafka-streams-0.10.2.0.jar:/data/bkce/service/kafka/b
```

```
in/..libs/kafka-streams-examples-0.10.2.0.jar:/data/bkce/service/kafka/bin/..libs/kaf
ka-tools-0.10.2.0.jar:/data/bkce/service/kafka/bin/..libs/log4j-1.2.17.jar:/data/bkce/
service/kafka/bin/..libs/lz4-1.3.0.jar:/data/bkce/service/kafka/bin/..libs/metrics-co
re-2.2.0.jar:/data/bkce/service/kafka/bin/..libs/reflections-0.9.10.jar:/data/bkce/service/kafka/bin/..
libs/rocksdbjni-5.0.1.jar:/data/bkce/service/kafka/bin/..libs/scala-library-2.12.1.jar
:/data/bkce/service/kafka/bin/..libs/scala-parser-combinators_2.12-1.0.4.jar:/data/bkc
e/service/kafka/bin/..libs/slf4j-api-1.7.21.jar:/data/bkce/service/kafka/bin/..libs/s
lf4j-log4j12-1.7.21.jar:/data/bkce/service/kafka/bin/..libs/snappy-java-1.1.2.6.jar:/d
ata/bkce/service/kafka/bin/..libs/validation-api-1.1.0.Final.jar:/data/bkce/service/ka
fka/bin/..libs/zkclient-0.10.jar:/data/bkce/service/kafka/bin/..libs/zookeeper-3.4.9.
jar kafka.Kafka ..config/server.properties
```

```
ps -ef | grep -i kafka
root      10220      1 2 Nov14 ?        13:17:44 /data/bkce/service/java/bin/java -Xmx1G
-Xms1G -server -XX:+UseG1GC -XX:MaxGCPauseMillis=20 -XX:InitiatingHeapOccupancyPercent=
35 -XX:+DisableExplicitGC -Djava.awt.headless=true -Xloggc:/data/bkce/logs/kafka/kafkaS
erver-gc.log -verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCTimeSta
mps -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.authenticate=false -D
com.sun.management.jmxremote.ssl=false -Dkafka.logs.dir=/data/bkce/logs/kafka -Dlog4j.c
onfiguration=file:../../../config/log4j.properties -cp /data/bkce/service/java/lib:/data/b
kce/service/kafka/bin/..libs/aopalliance-repackaged-2.5.0-b05.jar:/data/bkce/service/k
afka/bin/..libs/argparse4j-0.7.0.jar:/data/bkce/service/kafka/bin/..libs/connect-api-
0.10.2.0.jar:/data/bkce/service/kafka/bin/..libs/connect-file-0.10.2.0.jar:/data/bkce/
service/kafka/bin/..libs/connect-json-0.10.2.0.jar:/data/bkce/service/kafka/bin/..lib
s/connect-runtime-0.10.2.0.jar:/data/bkce/service/kafka/bin/..libs/connect-transforms-
0.10.2.0.jar:/data/bkce/service/kafka/bin/..libs/guava-18.0.jar:/data/bkce/service/kaf
ka/bin/..libs/hk2-api-2.5.0-b05.jar:/data/bkce/service/kafka/bin/..libs/hk2-locator-2
.5.0-b05.jar:/data/bkce/service/kafka/bin/..libs/hk2-utils-2.5.0-b05.jar:/data/bkce/se
rvice/kafka/bin/..libs/jackson-annotations-2.8.0.jar:/data/bkce/service/kafka/bin/..l
ibs/jackson-annotations-2.8.5.jar:/data/bkce/service/kafka/bin/..libs/jackson-core-2.8
.5.jar:/data/bkce/service/kafka/bin/..libs/jackson-databind-2.8.5.jar:/data/bkce/servi
ce/kafka/bin/..libs/jackson-jaxrs-base-2.8.5.jar:/data/bkce/service/kafka/bin/..libs/
jackson-jaxrs-json-provider-2.8.5.jar:/data/bkce/service/kafka/bin/..libs/jackson-modu
le-jaxb-annotations-2.8.5.jar:/data/bkce/service/kafka/bin/..libs/javassist-3.20.0-GA.
jar:/data/bkce/service/kafka/bin/..libs/javax.annotation-api-1.2.jar:/data/bkce/servic
e/kafka/bin/..libs/javax.inject-1.jar:/data/bkce/service/kafka/bin/..libs/javax.injec
t-2.5.0-b05.jar:/data/bkce/service/kafka/bin/..libs/javax.servlet-api-3.1.0.jar:/data/
bkce/service/kafka/bin/..libs/javax.ws.rs-api-2.0.1.jar:/data/bkce/service/kafka/bin/..
libs/jersey-client-2.24.jar:/data/bkce/service/kafka/bin/..libs/jersey-common-2.24.j
ar:/data/bkce/service/kafka/bin/..libs/jersey-container-servlet-2.24.jar:/data/bkce/se
rvice/kafka/bin/..libs/jersey-container-servlet-core-2.24.jar:/data/bkce/service/kaf
ka/bin/..libs/jersey-guava-2.24.jar:/data/bkce/service/kafka/bin/..libs/jersey-media-ja
xb-2.24.jar:/data/bkce/service/kafka/bin/..libs/jetty-continuation-9.2.15.v20160210.ja
r:/data/bkce/service/kafka/bin/..libs/jetty-http-9.2.15.v20160210.jar:/data/bkce/service/kaf
ka/bin/..libs/jetty-io-9.2.15.v20160210.jar:/data/bkce/service/kafka/bin/..libs/jetty-securit
y-9.2.15.v20160210.jar:/data/bkce/service/kafka/bin/..libs/jetty-server-9.2.15.v20160210.ja
r:/data/bkce/service/kafka/bin/..libs/jetty-server-9.2.15.v20160210.jar:/data/bkce/service/ka
fka/bin/..libs/jetty-servlet-9.2.15.v20160210.jar:/data/bkce/service/kafka/bin/..lib
s/jetty-util-9.2.15.v20160210.jar:/data/bkce/service/kafka/bin/..libs/jopt-simple-5.0
.3.jar:/data/bkce/service/kafka/bin/..libs/kafka_2.12-0.10.2.0.jar:/data/bkce/service/k
afka/bin/..libs/kafka_2.12-0.10.2.0-sources.jar:/data/bkce/service/kafka/bin/..libs/k
afka_2.12-0.10.2.0-test-sources.jar:/data/bkce/service/kafka/bin/..libs/kafka-clients-
0.10.2.0.jar:/data/bkce/service/kafka/bin/..libs/kafka-log4j-appender-0.10.2.0.jar:/da
ta/bkce/service/kafka/bin/..libs/kafka-streams-0.10.2.0.jar:/data/bkce/service/kafka/b
in/..libs/kafka-streams-examples-0.10.2.0.jar:/data/bkce/service/kafka/bin/..libs/kaf
ka-tools-0.10.2.0.jar:/data/bkce/service/kafka/bin/..libs/log4j-1.2.17.jar:/data/bkce/
service/kafka/bin/..libs/lz4-1.3.0.jar:/data/bkce/service/kafka/bin/..libs/metrics-co
re-2.2.0.jar:/data/bkce/service/kafka/bin/..libs/osgi-resource-locator-1.0.1.jar:/data
/bkce/service/kafka/bin/..libs/reflections-0.9.10.jar:/data/bkce/service/kafka/bin/..
libs/rocksdbjni-5.0.1.jar:/data/bkce/service/kafka/bin/..libs/scala-library-2.12.1.jar
:/data/bkce/service/kafka/bin/..libs/scala-parser-combinators_2.12-1.0.4.jar:/data/bkc
e/service/kafka/bin/..libs/slf4j-api-1.7.21.jar:/data/bkce/service/kafka/bin/..libs/
```

```
lf4j-log4j12-1.7.21.jar:/data/bkce/service/kafka/bin/../libs/snappy-java-1.1.2.6.jar:/data/bkce/service/kafka/bin/../libs/validation-api-1.1.0.Final.jar:/data/bkce/service/kafka/bin/../libs/zkclient-0.10.jar:/data/bkce/service/kafka/bin/../libs/zookeeper-3.4.9.jar kafka.Kafka ./config/server.properties
```

进程的功能名称均为 `java`，`启动参数匹配规则` 中分别输入 `zookeeper`、`kafka` 即可。

## 管理数据库/中间件实例：以 MySQL 为例

### 情景

应用使用的存储是 MySQL，为了便于 MySQL 的日常维护（如 SQL 变更），需要在 CMDB 中创建 MySQL CI 对象，录入 MySQL 实例。

“

蓝鲸 CMDB 拥有灵活的 CI 能力，掌握该教程后，可以管理数据库、中间件、硬件等 CI 对象

”

### 前提条件

在配置平台中 [新建好业务](#)，并 定义拓扑及分配主机。

**术语解释 - CI** : (Configuration Items)，资源对象，如 `MySQL`、主机、交易系统、交换机、路由器等  
**- CI 属性** : (Configuration Items Attribute)，资源对象的配置属性，如 MySQL CI 属性为实例名、IP、端口、存储引擎、数据库版本等  
**- CI 实例** : CI 的实例化，唯一识别一个资源对象，如 MySQL CI 实例为 `gd_area_master_01`

### 操作步骤

- 梳理：梳理 MySQL CI 属性
- 建模：创建 MySQL CI 对象
- 实例化：添加 MySQL 实例

#### 梳理 MySQL CI 属性

根据消费 MySQL 的场景，梳理 MySQL 常见的 CI 属性。

字段分组	字段中文名	字段英文名	字段类型	录入方式	是否唯一	是否必填
	实例名	bk_inst_name	短字符	自动	是	是

基本信息	IP地址	ip_addr	短字符	自动	是	是
	端口	port	数字	自动	是	是
	数据库版本	version	短字符	自动		
	安装路径	install_dir	长字符	自动		
	数据库文件路径	dbfile_dir	长字符	自动		
	是否开启binlog	enable_binlog	枚举	自动		
	是否开启慢查询日志	enable_slowlog	枚举	自动		
	字符集	chart_set	短字符	自动		
	存储引擎	storage_engine	短字符	自动		
	数据库大小	db_size	数字	自动		
核心参数	innodb缓存池大小	innodb_buffer_pool_size	数字	自动		
	innodb日志缓存大小	innodb_log_buffer_size	数字	自动		
	innodb日志磁盘写入策略	innodb_flush_log_at_trx_commit	短字符	自动		
	线程缓存大小	thread_cache_size	数字	自动		
	查询缓存大小	query_cache_size	数字	自动		
	最大连接数	max_connections	数字	自动		

其中，CI 属性自动录入参考[自动发现 MySQL 实例](#)。

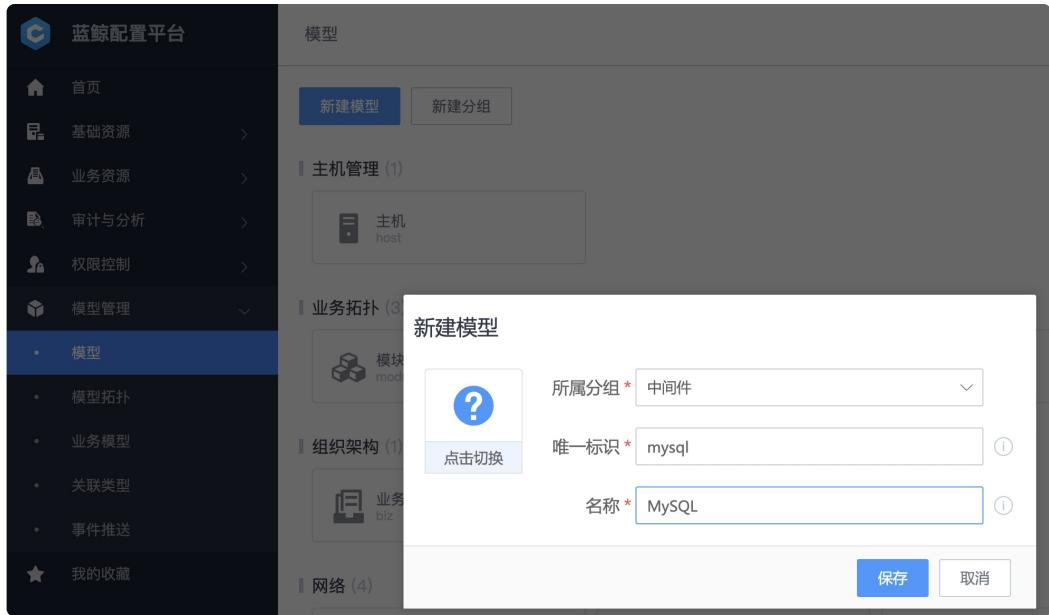
通过 **实例名** 可以唯一标识一个 MySQL 实例，具体是 **IP** 和 **端口** 的组合。

## 创建 MySQL CI 对象

第一步梳理完 MySQL CI 属性后，接下来开始建模：创建 MySQL CI。

### 创建 CI

选择 **模型管理** 中选择 **模型**，创建 MySQL 模型。



## 新增 CI 属性

按照梳理 MySQL CI 属性中梳理的结果来添加 CI 属性。

唯一标识	字段类型	必填	名称	创建时间	操作
blk_int_name	短字符串	✓	实例名	2019-07-18 17:58:46	<a href="#">编辑</a> <a href="#">删除</a>
character	字符串		字符集	2019-07-18 17:59:11	<a href="#">编辑</a> <a href="#">删除</a>
max_connections	数字		最大连接数	2019-07-18 17:59:11	<a href="#">编辑</a> <a href="#">删除</a>
db_version	短字符串		数据库版本	2019-07-18 17:59:11	<a href="#">编辑</a> <a href="#">删除</a>
install_dir	短字符串		安装路径	2019-07-18 17:59:11	<a href="#">编辑</a> <a href="#">删除</a>
dbtis_dir	短字符串		数据文件路径	2019-07-18 17:59:11	<a href="#">编辑</a> <a href="#">删除</a>
port	数字	✓	端口	2019-07-18 17:59:11	<a href="#">编辑</a> <a href="#">删除</a>
db_size	数字		数据库大小	2019-07-18 17:59:11	<a href="#">编辑</a> <a href="#">删除</a>
innodb_log_buff_size	数字		innodb日志缓冲大小	2019-07-18 17:59:11	<a href="#">编辑</a> <a href="#">删除</a>
thread_cache_size	数字		线程缓存大小	2019-07-18 17:59:11	<a href="#">编辑</a> <a href="#">删除</a>
query_cache_size	数字		查询缓存大小	2019-07-18 17:59:11	<a href="#">编辑</a> <a href="#">删除</a>
enable_binlog	枚举		是否开启binlog	2019-07-18 17:59:11	<a href="#">编辑</a> <a href="#">删除</a>
enable_slowlog	枚举		是否开启慢查日志	2019-07-18 17:59:11	<a href="#">编辑</a> <a href="#">删除</a>
storage_engine	短字符串		存储引擎	2019-07-18 17:59:11	<a href="#">编辑</a> <a href="#">删除</a>
innodb_log_pool_size	数字		innodb缓存大小	2019-07-18 17:59:11	<a href="#">编辑</a> <a href="#">删除</a>
innodb_flush_log_trx	枚举		innodb日志磁盘写入策略	2019-07-18 17:59:11	<a href="#">编辑</a> <a href="#">删除</a>
ip_addr	短字符串	✓	IP	2019-07-18 17:59:11	<a href="#">编辑</a> <a href="#">删除</a>

如果手工操作繁琐，也可以导入一个 MySQL 模型示例。

## 设立 CI 关联

针对 MySQL 的管理，除了 CI 属性外，我们同时还关心 MySQL 运行在哪台主机上，所以需要在模型中新建一个“主机上运行 MySQL”的关联。

1 个主机可以运行多个 MySQL 实例，所以 **源到目标的约束条件** 为“1-N”

唯一标识: mysql 名称: MySQL

模型字段 模型关联 唯一校验 字段分组

新建关联

唯一标识	关联类型	源-目标约束
------	------	--------

关联描述  
运行

源模型 \*  
主机

目标模型 \*  
MySQL

关联类型 \*  
run(运行)

源-目标约束 \*  
1-N

确定 取消

## 新增唯一校验

通过 实例名 可以唯一标识一个 MySQL 实例，具体是 IP 和 端口 的组合，所以将 IP 和端口作为一个组合校验。



唯一标识: mysql 名称: MySQL

模型字段 模型关联 唯一校验 字段分组

新建校验

校验规则	是否为必校验
实例名	是

校验规则 \*  
IP,端口

是否为必校验  
 是  否

确定 取消

## 添加 MySQL 实例

完成 MySQL CI 的建模之后，接下来添加 MySQL 实例

### 新增或导入 CI 实例

从首页进入 MySQL 实例列表页

The screenshot shows the BlueKing Configuration Platform homepage. On the left sidebar, under the 'MySQL' category, there is a red-bordered '新建' (New) button. The main content area displays a search bar with placeholder text '请输入IP开始查询主机...' and a search icon. Below the search bar is a table titled 'MySQL' with a count of 2 instances. The table has four columns: '主机管理 (2)' (Host Management), '网络 (4)' (Network), '中间件 (4)' (Middleware), and '应用 (1)' (Application). In the 'MySQL' row under '主机管理', there is another red-bordered '新建' button.

点击 **新建** 按钮，按提示添加 MySQL 实例，也可以批量导入 MySQL 实例。

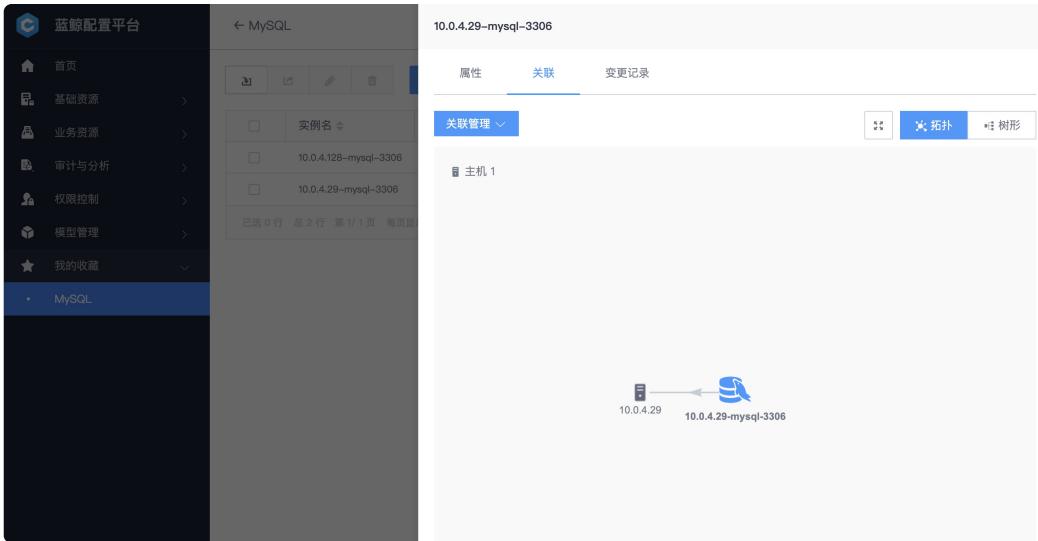
The screenshot shows the MySQL instance list page. The left sidebar has a red-bordered 'MySQL' category. The main area has a title 'MySQL' with a back arrow. It features a toolbar with icons for search, filter, and other actions, and a '新建' (New) button. There is a search bar with '实例名' (Instance Name) and '快速查询' (Quick Search) fields, along with a magnifying glass icon. A table lists two MySQL instances: '10.0.4.128-mysql-3306' and '10.0.4.29-mysql-3306'. Each entry includes columns for '实例名' (Instance Name), '端口' (Port), 'IP', '字符集' (Character Set), '最大连接数' (Max Connections), and '数据库版本' (Database Version). At the bottom, there are pagination controls showing '已选 0 行 总 2 行 第 1/1 页 每页显示 10 行'.

## 创建 CI 实例的关联关系

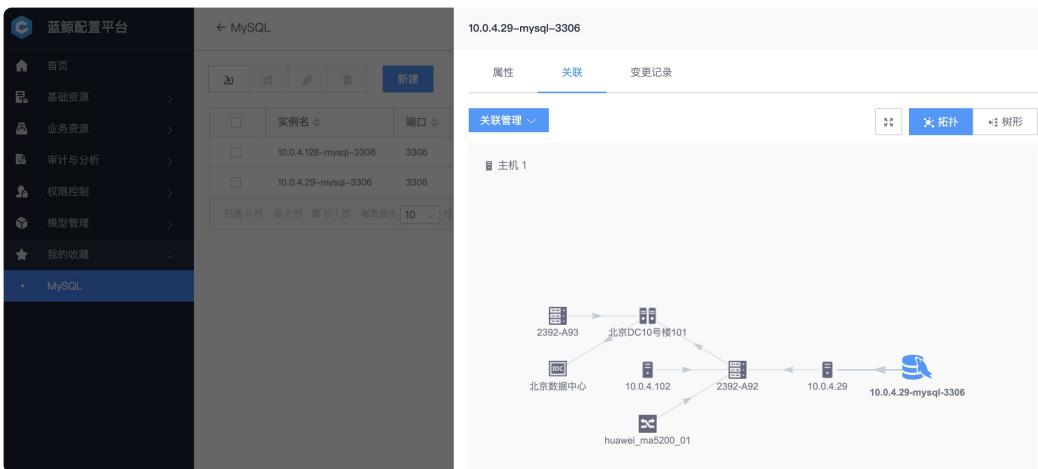
打开一个 MySQL 实例的详情页，点击 **关联** TAB 中的 **关联管理**，**关联** 当前实例运行在哪台主机上。

The screenshot shows the MySQL instance detail page for '10.0.4.128-mysql-3306'. The left sidebar has a red-bordered 'MySQL' category. The main area has a title '10.0.4.128-mysql-3306' with a back arrow. It features a toolbar with icons for edit, delete, and other actions, and a '新建' (New) button. The '关联' (Association) tab is selected. The '关联管理' (Association Management) section contains a table with columns 'ID', '内网IP' (Internal IP), and '操作' (Operation). The table lists 10 associations, each with an '关联' (Associate) link. At the bottom, there are pagination controls showing '总 26 行 第 1/3 页 每页显示 10 行'.

再次点击 **关联管理**，可预览 MySQL 实例与主机的关联关系。



如果参照本篇教程将 **机架**、**交换机**、**机房管理单元**、**数据中心** 等 IT 基础设施均录入 CMDB 中，将可以查询一个 MySQL 实例完整的关联关系。



## 自动发现数据库实例：以 MySQL 为例

“

感谢 **嘉为科技** 贡献该文档。

”

### 情景

业务存储层是 MySQL，通过蓝鲸 CMDB 已管理 MySQL 实例，然而却是手工维护，难以长期维护。需要通过自动化的方式实现，自动发现 MySQL 实例、MySQL CI 属性以及关联关系。

## 前提条件

1. 提前导入或录入 MySQL CI、创建模型（不是实例）的关联关系，详见：[如何管理数据库实例：以 MySQL 为例](#)
2. 在蓝鲸开发者中心 [新建一个应用](#)，用于调用 CMDB 的 API
3. 提供一个运维权限的账号以及 CMDB 的主机 IP 和端口，用于执行 JOB 作业
4. 创建查询 MySQL CI 属性的账号

```
sudo mysql -e "CREATE USER 'bk'@'{YOUR_MYSQL_IP}' IDENTIFIED BY '{PASSWORD_FOR_BK_QUERY}';"
sudo mysql -e "GRANT PROCESS, REPLICATION CLIENT ON *.* TO 'bk'@'{YOUR_MYSQL_IP}' WITH MAX_USER_CONNECTIONS 5;" 
sudo mysql -e "GRANT SELECT ON performance_schema.* TO 'bk'@'{YOUR_MYSQL_IP}';"
```

## 操作步骤

- 梳理自动发现逻辑
- 代码解读（含样例脚本）
- 测试效果

### 视频教程

{% video %}media/cmdb\_mysql\_autodiscovery.mp4{% endvideo %}

### 梳理自动发现逻辑

- **自动发现 MySQL 实例：**调用作业平台的快速脚本执行 API，在 MySQL 所在的主机上执行 ps 命令发现 MySQL 实例，调用 CMDB 的 API 创建实例。
- **自动发现 MySQL CI 属性：**调用 CMDB 获取实例的 API，获取 MySQL 实例所在的主机，调用作业平台的快速脚本执行 API，执行 SQL 语句查询 MySQL 属性，并调用 CMDB API 更新实例。
- **自动发现 MySQL 与所在主机的关联关系：**调用 CMDB 查询实例 API 找出 MySQL 实例对应主机，创建关联关系。

### 代码解读（含样例脚本）

#### 自动发现 MySQL 实例

- 通过 ps 命令获取 mysql 的 进程ID
- 通过 netstat 命令，根据 进程ID 获得 mysql 端口 号
- 调用 CMDB 创建实例的接口，将发现到的实例存入 CMDB

```
#!/usr/bin/env python
# -*- coding:utf-8 -*-

import requests
import json
import time
import base64
import sys
import warnings

warnings.filterwarnings("ignore")
reload(sys)
sys.setdefaultencoding('utf8')

BK_PAAS_HOST = 'http://'    # 蓝鲸 PaaS 地址, 例如https://paas.blueking.com/
BK_APP_CODE = ''    # 用于调用蓝鲸CMDB API 的 SaaS账号
BK_APP_TOKEN = ''    # 用于调用蓝鲸CMDB API 的 SaaS Token
BK_USERNAME = ''    # 拥有业务运维权限的蓝鲸账号
BK_BIZ_ID = ''    # 业务 ID, 在哪个业务下发现 MySQL 实例
BK_OBJ_ID = 'mysql'    # MySQL CI 名
BK_CMDB_HOST = 'http://{CMDB_HOST}:33031'    # 在蓝鲸业务下查看配置平台的主机IP或在install.conf中 cmdb 对应的 IP
MYSQL_HOST_IP = ['']    # 此处最好自行编写代码来获取 MySQL的IP
MYSQL_USERNAME = 'bk'    # 查询 MySQL CI 属性的 MySQL 账号
MYSQL_PASSWORD = ''    # 查询 MySQL CI 属性的 MySQL 密码
MYSQL_OS_ACCOUNT = 'root'    # 执行 JOB 作业的操作系统账号, 用于发现 MySQL 实例、CI属性、关联关系

# 自动发现MySQL
class CoverMysql(object):
    def __init__(self):
        self.bk_biz_id = BK_BIZ_ID
        self.job = JobMan()
        self.cover_ips = MYSQL_HOST_IP

    def start_cover(self):
        ip_list = self.search_ip_list()
        if not ip_list.__len__():
            return
        params = {
            "bk_biz_id": self.bk_biz_id,
            "script_content": self._mysql_script(),
            "script_timeout": 1000,
            "account": MYSQL_OS_ACCOUNT,
            "script_type": 1,
            "ip_list": ip_list,
        }
        self.job.fast_execute_sql(params)
        for x in ip_list:
            res, log_content = self.job.get_log(x['ip'])
            if not res:
                continue
            proc = json.loads(log_content)
            data = self._make_inst_data(x['ip'], proc)
            self.create_inst(data)

    def create_inst(self, data):
        url = BK_PAAS_HOST + '/api/c/compapi/v2/cc/create_inst/'
        params = {
            "bk_app_code": BK_APP_CODE,
            "bk_app_secret": BK_APP_TOKEN,
            "bk_username": BK_USERNAME,
            "bk_obj_id": BK_OBJ_ID,
            "bk_supplier_account": "0"
        }
```

```

        params = dict(params, **data)
        rp = requests.post(url, json.dumps(params), verify=False)
        if rp.status_code != 200:
            raise Exception(u'请求失败')

# 查询主机
def search_ip_list(self):
    url = BK_PAAS_HOST + '/api/c/compapi/v2/cc/search_host/'
    params = {
        "bk_app_code": BK_APP_CODE,
        "bk_app_secret": BK_APP_TOKEN,
        "bk_username": BK_USERNAME,
        "bk_supplier_account": "0",
        "bk_biz_id": self.bk_biz_id,
        "condition": [
            {
                "bk_obj_id": "host",
                "fields": [],
                "condition": [
                    {
                        "operator": '$eq',
                        "field": 'bk_os_type',
                        "value": '1' # 查询linux主机
                    },
                    {
                        "operator": '$in',
                        "field": 'bk_host_innerip',
                        "value": self.cover_ips
                    }
                ]
            },
            ],
        }
    rp = requests.post(url, json.dumps(params), verify=False)
    if rp.status_code != 200:
        raise Exception(u'请求失败')
    res = json.loads(rp.content)
    if not res['result']:
        raise Exception(u'查询主机失败')
    info = res['data']['info']
    return [
        {"bk_cloud_id": x['host']['bk_cloud_id'][0]['bk_inst_id'] if x['host']['bk_cloud_id'].__len__() else 0,
         "ip": x['host']['bk_host_innerip']}
    ] for x in info]

def _make_inst_data(self, ip, proc):
    port = proc.get('port', '')
    inst_name = '{0}-{1}-{2}'.format(ip, BK_OBJ_ID, port)
    return {
        'ip_addr': ip,
        'bk_inst_name': inst_name,
        'port': port
    }

def _mysql_script(self):
    script = u"""#!/bin/bash
# 获取进程端口号
Get_Port_Join_Str(){
    port_arr_str=$(netstat -ntlp | grep $1 | awk '{print $4}'|awk -F ':' '{print $NF}'|sed 's/ *$/g'|sed 's/^ *//g'|sort|uniq)
    if [ -z "$port_arr_str" ];then
        continue
    fi
    port_str="""

```

```

for port in ${port_arr[@]}
do
    if [ -n "$port_str" ];then
        port_str=${port_str},${port}
    else
        port_str=${port}
    fi
done
}

Get_Soft_Pid(){
i=0
soft_pid=()
pid_arr=$(ps -ef | grep -v grep | grep -i $1 | awk '{print $2}' )
for pid in ${pid_arr[@]}
do
    # 过滤掉端口不存在的进程
    port_str=$(netstat -ntlp | grep -w $pid)
    if [ -z "$port_str" ];then
        continue
    fi
    # 过滤掉不是mysql的进程
    is_mysql=$(($(readlink /proc/$pid/exe) -V 2>/dev/null|grep -i mysql)
    if [ -z "$is_mysql" ];then
        continue
    fi
    # 过滤掉蓝鲸sass 进程
    userId=$(ps -ef | grep $1 | grep -w $pid | grep -v grep | awk '{print $1}')
    if [[ "$userId" == "apps" ]];then
        continue
    fi
    # 筛选后的pid
    soft_pid[$i]=$pid
    i=$((expr $i + 1))
done
}

Cover_Mysql(){
condition=mysql
Get_Soft_Pid $condition
for pid in ${soft_pid[@]}
do
    Get_Port_Join_Str $pid
    exe_path=$(readlink /proc/$pid/exe)
    echo {'''port''': '''$port_str'''}
done
}
Cover_Mysql
"""

return base64.b64encode(script)

```

## 自动采集 MYSQL CI 属性

- 获取 CMDB 中 MySQL 的实例
- 根据 MySQL 中的 IP，调用作业平台执行脚本，采集配置信息
- 调用 CMDB 更新实例接口，将采集到的配置信息保存到 CMDB 中

```

class CollectMysql(object):
    def __init__(self):
        self.job = JobMan()

```

```

        self.cred = {
            'username': MYSQL_USERNAME,
            'password': MYSQL_PASSWORD
        }

    def start_collect(self):
        insts = self.search_inst()
        for inst in insts:
            hosts = self.search_host(inst['ip_addr'])
            if not hosts.__len__():
                print u'主机不存在'
                continue
            host = hosts[0]
            biz_id = host['biz'][0]['bk_biz_id']
            innerip = host['host']['bk_host_innerip']
            cloud_id = host['host']['bk_cloud_id'][0]['bk_inst_id']
            ip_list = [
                {
                    'ip': innerip,
                    'bk_cloud_id': cloud_id
                }
            ]
            params = {
                "bk_biz_id": biz_id,
                "script_content": self._mysql_script(innerip, self.cred['username'], self.cred['password']),
                "script_timeout": 1000,
                "account": MYSQL_OS_ACCOUNT,
                "script_type": 1,
                "ip_list": ip_list,
            }
            self.job.fast_execute_sql(params)
            res, log_content = self.job.get_log(innerip)
            if not res:
                continue
            inst_id = inst['bk_inst_id']
            config = json.loads(log_content)
            config = self.clear_data(config)
            self.update_inst(inst_id, config)
            asst_host = {
                'bk_asst_inst_id': inst_id,
                'bk_inst_id': host['host']['bk_host_id'],
                'bk_obj_asst_id': "host_run_{}".format(BK_OBJ_ID)
            }
            self.create_asst(asst_host)

    def search_inst(self):
        params = {
            "bk_app_code": BK_APP_ID,
            "bk_app_secret": BK_APP_TOKEN,
            "bk_username": BK_USERNAME,
            "bk_supplier_account": "0",
            "bk_obj_id": BK_OBJ_ID,
            "condition": {
                BK_OBJ_ID: []
            }
        }
        url = BK_PAAS_HOST + '/api/c/compapi/v2/cc/search_inst/'
        rp = requests.post(url, json.dumps(params), verify=False)
        if rp.status_code != 200:
            raise Exception(u'请求失败')
        res = json.loads(rp.content)
        if not res['result']:
            raise Exception(u'查询实例失败')
        return res['data']['info']

```

```

# 查询主机
def search_host(self, ip):
    url = BK_PAAS_HOST + '/api/c/compapi/v2/cc/search_host/'
    params = {
        "bk_app_code": BK_APP_ID,
        "bk_app_secret": BK_APP_TOKEN,
        "bk_username": BK_USERNAME,
        "bk_supplier_account": "0",
        "condition": [
            {
                "bk_obj_id": "host",
                "fields": [],
                "condition": [
                    {
                        "operator": '$in',
                        "field": 'bk_host_innerip',
                        "value": [ip]
                    }
                ]
            },
            {
                "bk_obj_id": "biz",
                "fields": [],
                "condition": []
            },
        ],
    }
    rp = requests.post(url, json.dumps(params), verify=False)
    if rp.status_code != 200:
        raise Exception(u'请求失败')
    res = json.loads(rp.content)
    if not res['result']:
        raise Exception(u'查询主机失败')
    return res['data']['info']

def update_inst(self, inst_id, config):
    params = {
        "bk_app_code": BK_APP_ID,
        "bk_app_secret": BK_APP_TOKEN,
        "bk_username": BK_USERNAME,
        "bk_supplier_account": "0",
        "bk_obj_id": BK_OBJ_ID,
        "bk_inst_id": inst_id
    }
    params = dict(params, **config)
    url = BK_PAAS_HOST + '/api/c/compapi/v2/cc/update_inst/'
    rp = requests.post(url, json.dumps(params), verify=False)
    if rp.status_code != 200:
        raise Exception(u'请求失败')
    res = json.loads(rp.content)
    if not res['result']:
        raise Exception(u'更新实例失败')

def clear_data(self, config):
    print config
    return {
        'character': config.get('charset', ''),
        'db_version': config.get('db_version', ''),
        'db_size': int(float(config.get('db_size', 0).rstrip('MB').rstrip('mb'))),
        'install_dir': config.get('basedir', ''),
        'dbfile_dir': config.get('datafile_path', ''),
        'enable_binlog': 'enable_binlog' if config.get('is_binlog', None) == 'ON' else
    }

```

```

lse 'disable_binlog',
    'enable_slowlog': 'enable_slowlog' if config.get('is_slow_query_log', None)
== 'ON' else 'disable_slowlog',
    'storage_engine': config.get('storage_engine', ''),
    'inno_buff_pool_size': int(float(config.get('innodb_buffer_pool_size', 0))
/ 1024 / 1024),
    'inno_log_buff_size': int(float(config.get('innodb_log_buffer_size', 0)) /
1024 / 1024),
    'inno_flush_log_trx': config.get('innodb_flush_log_at trx_commit', ''),
    'thread_cache_size': int(float(config.get('thread_cache_size', 0)) / 1024 /
1024),
    'query_cache_size': int(float(config.get('query_cache_size', 0)) / 1024 / 1
024),
    'max_connections': int(float(config.get('max_connections', 0)))
}
}

def _mysql_script(self, ip, username, password):
    mysql_script = u"""#!/bin/bash
ip= {%
    raw %}{{ip}}{%
    endraw %
}
username={%
    raw %}{{username}}{%
    endraw %
}
password={%
    raw %}{{password}}{%
    endraw %
}
sql1='''show VARIABLES WHERE variable_name LIKE "character_set_database"
OR variable_name LIKE "slow_query_log"
OR variable_name LIKE "datadir"
or variable_name LIKE "basedir"
OR variable_name LIKE "version"
or variable_name LIKE "log_bin" '''
sql2='SHOW VARIABLES WHERE variable_name ="default_storage_engine"'
# 大小
sql3="select concat(round((sum(data_length)+sum(index_length))/1024/1024,2),'MB') as da
ta from information_schema.tables"
# 数据库名
base_info=$(mysql -u$username -h$ip -p$password -s -e "${sql1}" 2>/dev/null)
sql10='''SHOW VARIABLES WHERE variable_name LIKE "innodb_buffer_pool_size" OR variable_
name LIKE "innodb_log_buffer_size" OR variable_name LIKE "innodb_flush_log_at_trx_commi
t" OR variable_name LIKE "thread_cache_size" OR variable_name LIKE "query_cache_size" O
R variable_name LIKE "max_connections"'''
base_info2=$(mysql -u$username -h$ip -p$password -s -e "${sql10}" 2>/dev/null)

charset=`echo $base_info|awk -F ' ' '{print $4}'` 
db_version=`echo $base_info|awk -F ' ' '{print $12}'` 
db_size=$(mysql -u$username -h$ip -p$password -s -e "${sql3}" 2>/dev/null)
basedir=`echo $base_info|awk -F ' ' '{print $2}'` 
datafile_path=`echo $base_info|awk -F ' ' '{print $6}'` 
storage_engine=`echo $(mysql -u$username -h$ip -p$password -s -e "${sql2}" 2>/dev/null)
|awk -F ' ' '{print $2}'` 
is_binlog=`echo $base_info|awk -F ' ' '{print $8}'` 
is_slow_query_log=`echo $base_info|awk -F ' ' '{print $10}'` 

innodb_buffer_pool_size=`echo $base_info2|awk -F ' ' '{print $2}'` 
innodb_log_buffer_size=`echo $base_info2|awk -F ' ' '{print $6}'` 
innodb_flush_log_at_trx_commit=`echo $base_info2|awk -F ' ' '{print $4}'` 
thread_cache_size=`echo $base_info2|awk -F ' ' '{print $12}'` 
query_cache_size=`echo $base_info2|awk -F ' ' '{print $10}'` 
max_connections=`echo $base_info2|awk -F ' ' '{print $8}'` 

echo {'''charset''': '''$charset''', \
'''db_version''': '''$db_version''', \
'''db_size''': '''$db_size''', \
'''basedir''': '''$basedir''', \
'''datafile_path''': '''$datafile_path''', \
'''storage_engine''': '''$storage_engine''', \
'''is_binlog''': '''$is_binlog''', \
}

```

```

'''is_slow_query_log'''': '''$is_slow_query_log''', \
'''innodb_buffer_pool_size'''': '''$innodb_buffer_pool_size''', \
'''innodb_log_buffer_size'''': '''$innodb_log_buffer_size''', \
'''innodb_flush_log_at_trx_commit'''': '''$innodb_flush_log_at_trx_commit''', \
'''thread_cache_size'''': '''$thread_cache_size''', \
'''query_cache_size'''': '''$query_cache_size''', \
'''max_connections'''': '''$max_connections''' \
}
exit
"""

mysql_script = mysql_script \
    .replace('{% raw %}{{username}}{% endraw %}', username) \
    .replace('{% raw %}{{password}}{% endraw %}', password) \
    .replace('{% raw %}{{ip}}{% endraw %}', ip)
return base64.b64encode(mysql_script)

```

## 自动发现关联信息

- 通过 MySQL 实例中 IP 地址，找到与之对应的主机
- 调用 CMDB 创建关联的接口，添加主机和 MySQL 直接的关联关系

```

# 发现 MySQL 实例与所在主机的关联关系
# 创建关联: CMDB_HOST + '/api/v3/inst/association/action/create'
# 目前暂未开放ESB接口，接口地址和参数通过抓包获取
# params:
# {
#     'bk_asst_inst_id': 目标实例ID, * 必填
#     'bk_inst_id': 源实例ID, * 必填
#     'bk_obj_asst_id': 关联ID * 必填
# }
def create_asst(self, params):
    is_asst = self.search_asst(params) # 判断关联关系是否已存在
    if is_asst:
        return
    url = BK_CMDB_HOST + '/api/v3/inst/association/action/create'
    headers = {
        "Content-Type": "application/json",
        "HTTP_BLUEKING_SUPPLIER_ID": "0",
        "BK_USER": BK_USERNAME
    }
    rp = requests.post(url=url, data=json.dumps(params), headers=headers, verify=False)
    if rp.status_code != 200:
        raise Exception(u'请求失败')
    res = json.loads(rp.content)
    if not res['result']:
        print res
        raise Exception(u'创建关联失败')

# 查询关联: CMDB_HOST + '/api/v3/object/association/action/search'
# 目前暂未开放ESB接口，接口地址和参数通过抓包获取
# params:
# {
#     'bk_asst_inst_id': 目标实例ID,
#     'bk_inst_id': 源实例ID,
#     'bk_obj_id': 模型ID, * 必填
#     'bk_obj_asst_id': 关联ID
# }
def search_asst(self, params):
    url = BK_CMDB_HOST + '/api/v3/inst/association/action/search'
    params['bk_obj_id'] = BK_OBJ_ID

```

```

        headers = {
            "Content-Type": "application/json",
            "HTTP_BLUEKING_SUPPLIER_ID": "0",
            "BK_USER": BK_USERNAME
        }
        rp = requests.post(url=url, data=json.dumps(params), headers=headers, verify=False)
    else:
        if rp.status_code != 200:
            raise Exception(u'请求失败')
        res = json.loads(rp.content)
        if not res['result']:
            raise Exception(u'查询关联')
        for asst in res['data']:
            if params['bk_obj_asst_id'] == asst['bk_obj_asst_id'] and \
                params['bk_inst_id'] == asst['bk_inst_id'] and \
                params['bk_asst_inst_id'] == asst['bk_asst_inst_id']:
                return True
    return False

```

## 封装作业平台执行类

封装好调用作业平台快速执行脚本接口的类

```

class JobMan(object):
    def __init__(self):
        self.ip_log = {}

    def get_log(self, ip):
        if ip not in self.ip_log:
            return False, u"IP{0}未执行作业, 可能是IP未录入或Agent异常".format(ip)
        return self.ip_log[ip]["is_success"], self.ip_log[ip]["log_content"]

    # 快速执行脚本
    def fast_execute_sql(self, params):
        url = BK_PAAS_HOST + '/api/c/compapi/v2/job/fast_execute_script/'
        params = dict({
            "bk_app_code": BK_APP_ID,
            "bk_app_secret": BK_APP_TOKEN,
            "bk_username": BK_USERNAME,
        }, **params)
        rp = requests.post(url, json.dumps(params), verify=False)
        if rp.status_code != 200:
            raise Exception(u'请求失败')
        res = json.loads(rp.content)
        if not res['result']:
            raise Exception(u'执行脚本失败')
        self.get_job_instance_log(params['bk_biz_id'], res['data']['job_instance_id'])

    def get_job_instance_log(self, bk_biz_id, job_instance_id):
        params = {
            "bk_app_code": BK_APP_ID,
            "bk_app_secret": BK_APP_TOKEN,
            "bk_username": BK_USERNAME,
            "bk_biz_id": bk_biz_id,
            "job_instance_id": job_instance_id
        }
        url = BK_PAAS_HOST + '/api/c/compapi/v2/job/get_job_instance_log/'
        rp = requests.post(url, json.dumps(params), verify=False)
        if rp.status_code != 200:
            raise Exception(u'请求失败')
        res = json.loads(rp.content)
        if not res['result']:

```

```

        raise Exception(u'获取日志失败')
    step = res['data'][0]
    if step['is_finished']:
        self.__get_step_log(step)
    else:
        time.sleep(5)
        self.get_job_instance_log(bk_biz_id, job_instance_id)

def __get_step_log(self, step):
    for step_res in step['step_results']:
        for ip_content in step_res['ip_logs']:
            if ip_content['ip'] in self.ip_log:
                print ip_content['ip'] + u"IP重复"
                continue
            self.ip_log[ip_content['ip']] = {
                "is_success": step_res["ip_status"] == 9,
                "source": ip_content['bk_cloud_id'],
                "ip": ip_content['ip'],
                "log_content": ip_content['log_content']
}

```

## 调用执行

```

if __name__ == '__main__':
    # 自动发现
    cover = CoverMysql()
    cover.start_cover()
    # 自动采集
    collect = CollectMysql()
    collect.start_collect()

```

## 测试效果

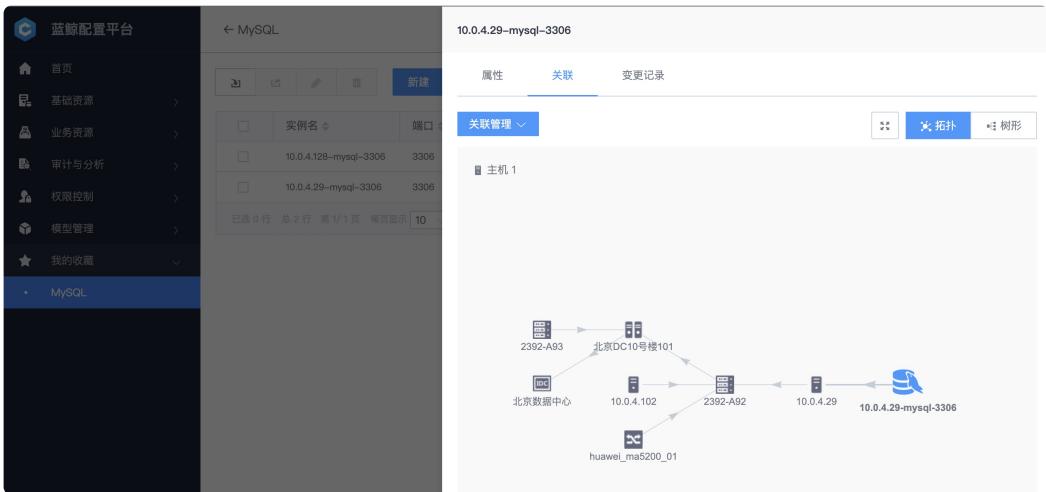
- 自动发现 MySQL 实例

	实例名	端口	IP	字符集	最大连接数	数据库版本
1	10.0.4.128-mysql-3306	3306	10.0.4.128	latin1	151	5.5.60-MariaDB
2	10.0.4.29-mysql-3306	3306	10.0.4.29	latin1	151	5.5.60-MariaDB

- 自动发现 MySQL CI 属性



- 自动发现 MySQL 与主机的关联关系



“

注：关联关系中，主机与机架、机房管理单元、机房等关联需要单独添加对应的关联关系，此处不做展开。

”

## MySQL 自动发现采集器 Demo

请 [下载 demo](#) 后，在一台有 Python 环境的主机（可访问蓝鲸）或 SaaS 中运行。

## 其他说明

要实现周期性自动发现，可用 [Celery](#) 的周期任务实现

# 企业原有 CMDB 同步至蓝鲸 CMDB

“

感谢社区用户 **Kevin** 提供该文档.

”

## 情景

公司基础部门维护公司统一的一套 CMDB，资源主要以服务器为主，资源交付后会直接进入公司的 CMDB，需要手动导出列表，修改成蓝鲸 CMDB 的导入格式，导入蓝鲸 CMDB。存在如下问题：

- 资源交付后需要手动导出、导入，少量机器频繁交付，为了让机器及时加入蓝鲸管控，需要频繁人工参与
- 公司 CMDB 与蓝鲸 CMDB 字段不统一，需要手动复制、编辑字段属性，再导入蓝鲸，非常繁琐

## 前提条件

- 在配置平台中 [新建好业务](#)。
- 熟悉一门脚本语言，如 [Python](#)，本教程以 [Python](#) 为例
- 了解 [蓝鲸 CMDB API](#)

## 操作步骤

- 梳理同步逻辑
- 代码实践及解读
- 定时同步

### 梳理同步逻辑

查询公司 CMDB 与蓝鲸 CMDB 服务器属性的对应关系，生成属性映射 MAP 表，MAP 表中只记录需要关注的属性，并确认相关属性都已在蓝鲸 CMDB 中创建。

#### 新增资源

- 通过接口查询公司 CMDB，获取自己所在业务部门下的全量服务器列表
- 通过接口获取蓝鲸 CMDB 的服务器列表
- 遍历两个列表，找出蓝鲸 CMDB 中没有的服务器，汇总成列表，并通过接口录入到蓝鲸 CMDB

## 更新资源

- 分别获取两个 CMDB 的数据，遍历两组数据，对比属性 MAP 表中的属性字段，找出数据不同的服务器，汇总成列表，再统一更新到蓝鲸 CMDB

## 代码实践及解读

### 新建应用

在蓝鲸开发者中心 [新建一个应用](#)，用于调用 CMDB 的 API。

```
import urllib2
import json
import logging

PAAS_API = 'https://paas.bk.com' # 蓝鲸 PaaS 地址

logging.basicConfig(level=logging.DEBUG,
    format='%(asctime)s %(filename)s[line:%(lineno)d] %(levelname)s %(message)s',
    datefmt='%a, %d %b %Y %H:%M:%S',
    filename='./update.log',
    filemode='a')

BASE_ARGS = {
    ...
    接口调用中的通用参数
    蓝鲸中新建一个应用，获取相关信息
    并在https://paas.bk.com/admin/bkcore/functioncontroller/中添加认证白名单，否则调用CMDB
    接口会失败，参考： 开发者中心-API网关-使用指南-API调用说明
    ...
    'bk_app_code': 'sync-cmdb', # 应用 ID
    'bk_app_secret': 'xxxx-xxxxxxxxxxxx', # 应用 Token
    'bk_username':'user.name' # 蓝鲸系统中的用户名
}
```

### 获取蓝鲸 CMDB 主机列表

查询蓝鲸 CMDB 的获取主机 `search_host` API，返回主机列表

```
def getBkCmdbHost():
    # 查询蓝鲸CMDB，返回主机列表
    bkList = {}
    url = PAAS_API + '/api/c/compapi/v2/cc/search_host/' # api请求地址
    apiArgs = {
        # api相关参数，参考https://paas.bk.com/esb/api\_docs/system/CC/search\_host/
        'condition':[
            {
                "bk_obj_id": "host",
                "fields": [],
                "condition": []
            }
        ]
    }

    args = dict(BASE_ARGS.items() + apiArgs.items())
    data = json.dumps(args)
    request = urllib2.Request(url, data=data)
    response = urllib2.urlopen(request)
    content = response.read()
```

```
j_content = json.loads(content)
for item in j_content['data']['info']:
    bkList[item['host']['bk_host_innerip']] = item['host']
return bkList
```

## 获取公司 CMDB 主机列表

调用公司 CMDB 获取主机列表 API，返回主机列表

```
def getCmdbHost():
    # 根据企业内部接口自己实现，返回IP信息列表
    ...
    infoList = [
        "ip1": {
            "sn": "xxxx",
            "service_term": 3,
            ...
        },
        "ip2": {},
        ...
    ]
    ...
```

## 生成新增 IP 列表、更新 IP 列表、更新信息列表

根据 [获取蓝鲸 CMDB 主机列表操作步骤](#) 和 [获取公司 CMDB 主机列表操作步骤](#) 获取到的两个主机列表，对比生成所需的新增列表和更新列表。

```
def genAddList(cmdbList, bkList):
    # 对比两个列表，返回新增IP列表
    addList = []
    for k in cmdbList.keys():
        if k not in bkList.keys():
            #print k, cmdbList[k]
            addList.append(k)
    return addList

def genUpdateList(cmdbList, bkList):
    # 对比两个列表，生成需要更新的IP列表
    ipList = []
    attrMAP = {k: v for k, v in attributeMAP.items() if v is not None} # 过滤出自己关心的
    主机属性
    for ip, cmdbHost in cmdbList.items():
        if ip not in bkList.keys():
            continue
        bkHost = bkList[ip]
        for cmdbAttr, bkAttr in attrMAP.items():
            if cmdbAttr in cmdbHost.keys() and bkAttr in bkHost.keys() and cmdbHost[cmd
            bAttr] != bkHost[bkAttr]:
                ipList.append(ip)
    return ipList

def genInfoList(cmdbList, bkList, ipList, attributeMAP):
    # 根据IP列表生成主机详细信息列表
    infoList = {}
    attrMAP = {k: v for k, v in attributeMAP.items() if v is not None}
    for ip in ipList:
        ipInfo = {}
        for k, v in attrMAP.items():
```

```

        if k in cmdbList[ip].keys():
            ipInfo[attrMAP[k]] = cmdbList[ip][k]
        if ip in bkList.keys(): #添加bk_host_id
            ipInfo['bk_host_id'] = bkList[ip]['bk_host_id']
        infoList[ip] = ipInfo
    return infoList

attributeMAP = {
    # 属性对应列表, 值为None的字段不关心, 不录入蓝鲸cmdb
    'AssetNo': 'bk_asset_id', #资产编号
    'SN': 'bk_sn', #序列号
    'Manufacture': 'machine_manufacturer', #厂商
    'ManufactureType': 'machine_model', #厂商型号
    'MachineType': 'machine_type', #MachineType
    'HostName': None, #主机名
    'IP': 'bk_host_innerip', #IP
    'BusinessIP': 'bk_host_bizip', #业务IP
    'PublicIP': 'bk_host_outerip', #公网IP
    'VNetIP': 'bk_host_usernetip', #用户网IP
    'tags': None, #资产类型
    #其它...
}

addList = genAddList(cmdbList, bkList)
updateList = genUpdateList(cmdbList, bkList):
addInfoList = genInfoList(cmdbList, bkList, addList, attributeMAP)
updateInfoList = genInfoList(cmdbList, bkList, addList, attributeMAP)

```

## 录入、更新主机数据到蓝鲸 CMDB

- 录入新增主机信息：使用 **生成新增 IP 列表、更新 IP 列表、更新信息列表** 操作步骤 中生成的新增主机信息列表，录入蓝鲸 CMDB

```

def addBkCmdbHost(infoList):
    # 录入新增数据到cmdb
    ...
    infoList = [
        "ip1": {
            "bk_sn": "xxxx",
            "bk_service_term": 3,
            ...
        },
        "ip2": {},
        ...
    ]
    ...

    ipInfoList = {}
    for k, v in enumerate(infoList.values()):
        v['bk_cloud_id'] = 0
        v['import_from'] = '3'
        if 'bk_host_id' in v.keys():
            del(v['bk_host_id'])
        #print k, v
        ipInfoList[str(k)] = v

    url = PAAS_API + '/api/c/compapi/v2/cc/add_host_to_resource/'
    apiArgs = {
        'host_info': ipInfoList
    }

```

```

args = dict(BASE_ARGS.items() + apiArgs.items())
logging.debug(args)
data = json.dumps(args)
request = urllib2.Request(url, data=data)
response = urllib2.urlopen(request)
content = response.read()
print content

addBkCmdbHost(addInfoList)

```

- 更新主机数据：使用 [生成新增 IP 列表](#)、[更新 IP 列表](#)、[更新信息列表](#)操作步骤 中生成的更新主机信息列表，更新蓝鲸 CMDB

```

def updateBkCmdbHost(infoList):
    # 更新数据到cmdb
    url = PAAS_API + '/api/c/compapi/v2/cc/update_host/'
    for ip, ipInfo in infoList.items():
        bk_host_id = ipInfo['bk_host_id']
        del(ipInfo['bk_host_id'])
        apiArgs = {
            'bk_host_id': bk_host_id,
            'data': ipInfo
        }

        #print apiArgs
        args = dict(BASE_ARGS.items() + apiArgs.items())
        logging.debug(args)
        data = json.dumps(args)
        request = urllib2.Request(url, data=data)
        response = urllib2.urlopen(request)
        content = response.read()
        logging.debug('bk_host_id: ' + str(bk_host_id) + content)
        print json.loads(content)

updateBkCmdbHost(updateInfoList)

```

## 定时同步

- CMDB 同步脚本使用 [JOB](#) 的定时作业或 [标准运维](#) 的定时任务进行周期托管，方便迁移或者修改维护

标准运维

流程模板 周期任务 任务记录 业务配置 轻应用 管理员入口

新建任务

节点选择

任务信息

任务名称 \* CMDB数据同步

执行计划 \* 立即执行 周期执行

周期表达式 \* 0 \* \* \* \*

参数信息

无数据

上一步 下一步

注：上述教程是企业 CMDB **单向**、**定期** 同步主机实例至蓝鲸 CMDB 的实践，如果需要实时同步，一般推荐 **消息推动** 的方式。

## Open-Falcon 集成蓝鲸 CMDB 业务拓扑树

“

感谢社区用户 **StephenWang** 贡献该文档。

”

### 情景

**Open-Falcon** 是一款开源监控产品， 默认通过 **Endpoint**（一般为主机名）查找服务器，体验不够友好。

集成蓝鲸配置平台的业务拓扑树，将首页左侧栏改造为服务树，通过选择业务模块快速选择机器，提高监控查看效率。

## 前提条件

- 主机纳管在蓝鲸配置平台中
- 在蓝鲸开发者中心 新建一个应用，用于调用 CMDB 的 API
- 熟悉 Python、JavaScript

## 操作步骤

- 配置平台中建立业务拓扑
- 查询主机及拓扑，写入 Redis
- 封装后台接口
- Open-Falcon 前端调用
- 预览效果

### 配置平台中建立业务拓扑

参照 [主机纳管在蓝鲸配置平台中](#)，根据应用的部署分层架构，建立业务拓扑如下：

内网IP	云区域	模块名	集群名字	外网IP
阿里云 [REDACTED]	空闲机	空闲机	空闲机池	--
上海世纪互联IDC [REDACTED]	pbi.bcs	pay	--	--
上海世纪互联IDC [REDACTED]	rkg.oas	frc	--	--
阿里云 [REDACTED]	trippublicserv	btc	--	--
上海世纪互联IDC [REDACTED]	fb2calc	fb5	--	--
上海世纪互联IDC [REDACTED]	jim	fmc	--	--
上海世纪互联IDC [REDACTED]	lmb.lmf	ivu	--	--
上海世纪互联IDC [REDACTED]	aflm.aft	frc	--	--
南京烽火IDC [REDACTED]	sdc.cbs	btc	--	--
上海世纪互联IDC [REDACTED]	prepaysettlements...	erp	--	--

### 查询主机及拓扑，写入 Redis

- 定时调用配置平台 [查询主机:search\\_host](#) 接口
- 将结果转化为 ztree 的数据格式，并写入 Redis

```
# -*- coding: utf-8 -*-
import requests
import redis
```

```

url = '/api/c/compapi/v2/cc/search_host/'
data = {
    "bk_app_code": "", # 用于调用蓝鲸CMDB API的SaaS账号
    "bk_app_secret": "", # 用于调用蓝鲸CMDB API的SaaS Token
    "bk_username": "", # 拥有业务运维权限的蓝鲸账号
    "condition": [
        {
            "bk_obj_id": "host",
            "fields": []
        },
        {
            "bk_obj_id": "module",
            "fields": []
        },
        {
            "bk_obj_id": "set",
            "fields": []
        },
        {
            "bk_obj_id": "biz",
            "fields": [],
            "condition": [
                {
                    "field": "bk_biz_id",
                    "operator": "$eq",
                    "value": 3
                }
            ]
        }
    ]
}

topo = []
result = requests.post(url=url, json=data).json()
for res in result['data']['info']:
    ip = res['host']['bk_host_innerip']
    for module in res['module']:
        topo2 = module['TopModuleName'].split('#')
        sett = topo2[1]
        modd = topo2[1] + '_' + topo2[2]
        if {'id': sett, 'pid': 0, 'name': sett} not in topo:
            topo.append({'id': sett, 'pid': 0, 'name': sett})
        if {'id': modd, 'pid': sett, 'name': modd} not in topo:
            topo.append({'id': modd, 'pid': sett, 'name': modd})
        if {'id': ip, 'pid': modd, 'name': ip} not in topo:
            topo.append({'id': ip, 'pid': modd, 'name': ip})

rdp = redis.ConnectionPool(host='', port=6379)
rdc = redis.StrictRedis(connection_pool=rdp)

topo = sorted(topo, key=lambda x: x['name'])
rdc.set('cmdb', str({'value': topo}))

```

## 封装后台接口

- 用 `flask` 开发接口，用于 OpenFalcon 前端调用
- 跨域访问处理：调用外部接口，需要解决跨域问题

```

#-*-coding:utf-8 -*-
import flask
import redis
from flask import jsonify
from flask import request
from flask import make_response

server = flask.Flask(__name__)
@server.route('/topo_new_ztree/',methods=['post'])

def search_new_ztree():
    """
    :method: post
    :return:
    """
    try:
        rdp = redis.ConnectionPool(host='', port=6379)
        rdc = redis.StrictRedis(connection_pool=rdp)
        topo = eval(rdc.get('cmdb'))["value"]
        response = make_response(jsonify({'result': 'success', 'message': topo}))
    except Exception as e:
        response = make_response(jsonify({'message': {'msg': str(e)}}, 'result': 'fail'))
    finally:
        response.headers['Access-Control-Allow-Origin'] = '*'
        response.headers['Access-Control-Allow-Methods'] = 'POST'
        response.headers['Access-Control-Allow-Headers'] = 'x-requested-with,content-type'
    return response

```

## 前端样式调整

修改 Open-Falcon 前端页面 `dashboard/rrd/templates/index.html`

前端拓扑树选择 `ztree` 插件：支持模糊查询，勾选事件等，具体功能可查询 [官方 API](#)

- 调用上述的接口，`ztree` 加载返回结果
- 拓扑树中可以自定义勾选事件

```

var post_url='/topo_new_ztree/';

function createTree(post_url){
    var zTree;
    var setting = {
        check: {
            enable: true
        },
        view: {
            dblClickExpand: true,
        },
        data: {
            simpleData: {
                enable: true,
                idKey: "id",
                pIdKey: "pid",
                rootPId: 0
            }
        }
    },

```

```

        callback:{  

            onCheck:onCheck  

        }  

    };  

    function onCheck(){  

        //自定义勾选事件  

    };  

$.post(post_url,function(data){  

    zTree = $.fn.zTree.init($("#tree"), setting, data.message);  

});  

}

```

## 预览效果

可直接通过左侧的服务树选择资源，十分方便。

The screenshot shows a monitoring system interface. On the left, there is a 'Service Tree' sidebar with a tree view of resources. The main area has two search panels: one for 'Endpoints' and one for 'Counters'. The 'Endpoints' panel includes a search bar, a 'Labels' section, and a 'Quick Filter' button. The 'Counters' panel includes a search bar, a 'Labels' section, and a table of counter metrics with columns for 'Counter', 'Type', and 'Step'.

改造前，查找资源只能通过检索，十分不便，如下图：

This screenshot shows the original monitoring system's search interface. It consists of two separate panels: '搜索Endpoints' (Search Endpoints) on the left and '搜索Counters' (Search Counters) on the right. Both panels have search bars, 'Labels' sections, and 'Quick Filter' buttons. The 'Counters' panel also includes a table of counter metrics with columns for 'Counter', 'Type', and 'Frequency'.



注：CMDB 中的资源经常变动，建议使用事件驱动的消息推送来监听资源变化，而不是周期获取。

”

# 应用交付，从人工到自动化的转变

“

在提升企业应用交付效率的同时，输出运维价值

”

**配置管理标准化** 提到，运维服务“四化建设”的标准化是如何规范运维在配置管理、发布、变更、故障处理、监控告警等场景的运维能力，重点阐述企业如何建设**配置管理标准化**。

接下来，我们通过理论配合实践（以**蓝鲸标准运维**为例），分析“四化建设”第2个阶段：**自动化**，如何让应用交付从人工到自动化的转变，在提升企业应用交付效率的同时，输出运维价值。

首先，我们看下应用交付遇到的痛点。

## 应用交付痛点

管理者觉得在质量、效率、成本、安全上你做得都很差，而作为执行者的你，心累、身体累的同时还担心哪天不小心背锅而走。

- 应用部署至生产环境周期长，无法快速迭代
- 部署过程中跨越多个运维系统，操作较为繁琐，容易出错
- 部署到生产环境后，可能面临回滚
- 无法直观查看部署进展，心里没底
- 每周大部分时间投入到发布、变更、故障处理，谈什么运维价值
- 运维操作零散，不可审计，无法满足合规性要求
- ...

## 项目目标

### 术语解释

- **资源编排**：(Resource Orchestration)，用户定义资源的编排流程，工具执行编排流程，执行时只关注输入，提升重复性应用/资源交付的运维效率。
- **持续集成[1]**：(Continues Integration)，Code -> 自动化测试 -> 合入主分支 -> 归档构建（最好是捕获 commit 事件自动触发 CI）

- **持续交付** : (Continues Delivery), 归档构建 -> 测试环境自动交付 ( CI 成功后, 会自动触发持续交付, 在同一个 pipeline 中)
- **持续部署** : (Continues Deployment), 归档构建 -> 生产环境自动部署 (灰度发布、全量发布等)

## 项目目标

在运维“四化建设”基石“**标准化**”的基础上, 从 **运维四要素** : **质量**、**效率**、**成本**、**安全** 的角度出发, 以实现应用交付自动化的目标。

- 标准化: 通过流程模板固化在标准化阶段沉淀的发布、变更、故障处理流程
- 运维四要素
  - 质量: 减少人工实施导致的人为错误
  - 效率: 应用交付提速, 从多天 1 发到 1 天多发
  - 成本: 认领工单后, 只需关注交付结果, 让运维更聚焦增值服务
  - 安全: 满足安全合规要求, 实现应用交付过程可视化, 交付历史归档可查



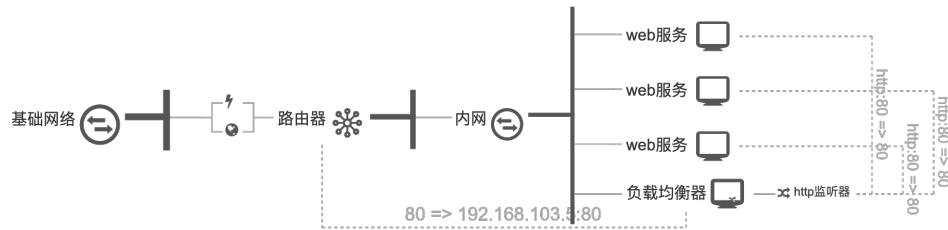
蓝鲸智云

## 技术选型

需要一套资源编排工具, 串接应用交付过程中的各个原子节点。如执行类: 文件分发、命令执行、DB 提单等, 周边类: 告警屏蔽、任务通知、审批等, 流程类: 认领 ITSM 单据, 结单等)。

放眼行业, 资源编排在公有云中早有对应的成熟工具[2]

行业应用	AWS	Azure	谷歌云	阿里云	青云
对应产品	Cloud Formation	Azure-RM	Cloud Deployment Manager	ROS	RO



(青云的资源编排工具)

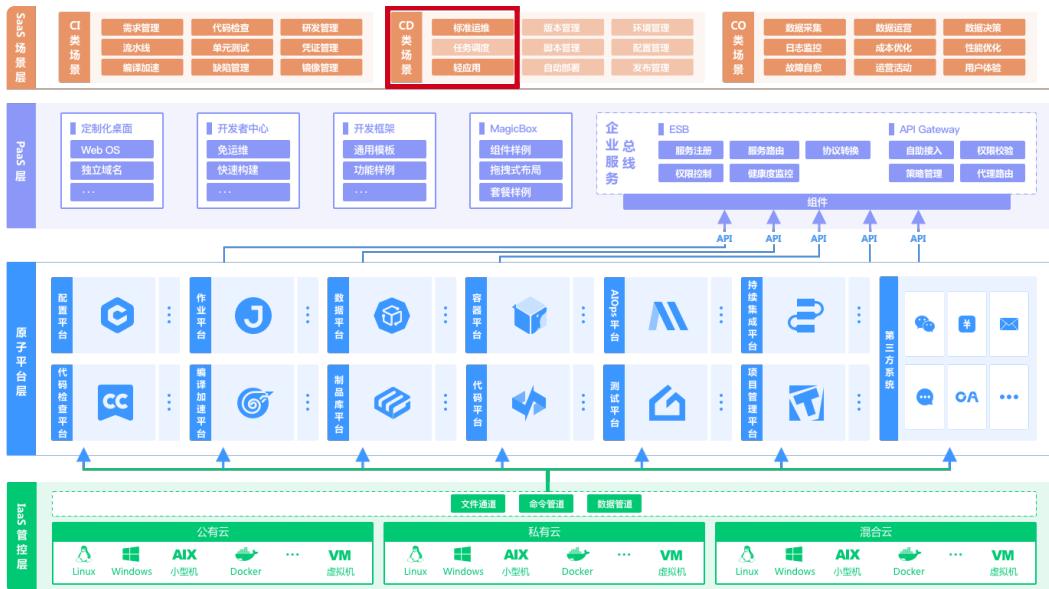
然而，行业慢慢关注混合云，从信通院发布的《混合云白皮书（2019年）》[3]中获悉，2018年国内混合云的占有率为8.1%，同期全球占比58%，以至于Terraform[4]慢慢成长起来了。

然而，这些资源编排工具只能满足企业在云端（公有云、私有云、混合云）资源的编排，包括Ansible在内的自动化工具无法满足应用交付过程中与企业IT系统的交互，比如发布工单、CMDB、审批流程等等，更多仍然还是偏执行。

蓝鲸标准运维，正是解决上述行业痛点。

流程编排服务	一键自动化调度	多元接入支持	助力业务自助化
遵循业界通用的流程建模标准BPMN2.0，提供可视化的流程编排服务，支持原子节点的串行、并行，支持子流程	基于蓝鲸PaaS平台API网关对接企业内部各个系统API的能力，将多系统间的工作整合到一个流程中，实现一键自动化调度	对接了蓝鲸通知、作业平台、配置平台等服务，还支持用户自定义快速接入企业内部系统	屏蔽了底层系统差异，让运维人员可以将业务日常的运维工作交给产品、开发、测试等人员执行，实现业务的发布、变更等工作自助化

标准运维在蓝鲸体系中位于SaaS场景层，通过PaaS层的ESB或API Gateway与原子平台层的CMDB、作业平台等平台交互。



确立了技术方案后，接下来介绍如何在企业内部实施标准运维，实现应用交付自动化。

如何实施标准运维，请参考场景案例：[应用如何自动化部署至生产环境、应用交付人力转移，让运维更专注业务优化。](#)

## 技术发展方向

蓝鲸体系内覆盖更多的场景，如告警屏蔽、解屏蔽，关联 ITSM 工单等，单一来源使其自成体系，体验更流畅。

丰富持续部署领域所依赖资源的标准插件，如各大公有云和私有云（Openstack 等）的资源（云主机、数据库、中间件等），向 Terraform 看齐。

支持模板语法（如 YAML、JSON、Python、Jinja 等），增强复杂场景的编排能力。

开放基于 **BPMN** 的画布设计器，不止与底层编排服务，让更多的场景 SaaS（如运营活动逻辑的自助配置）可以低成本的构建。

标准运维负责 DevOps 流程的 CD（持续部署）部分，将版本部署至生产环境，而 [蓝盾 CI 平台](#) 负责之前的 CI（持续集成）+ CD（持续交付），二者间的联动，让整个 DevOps 流程更加流畅。

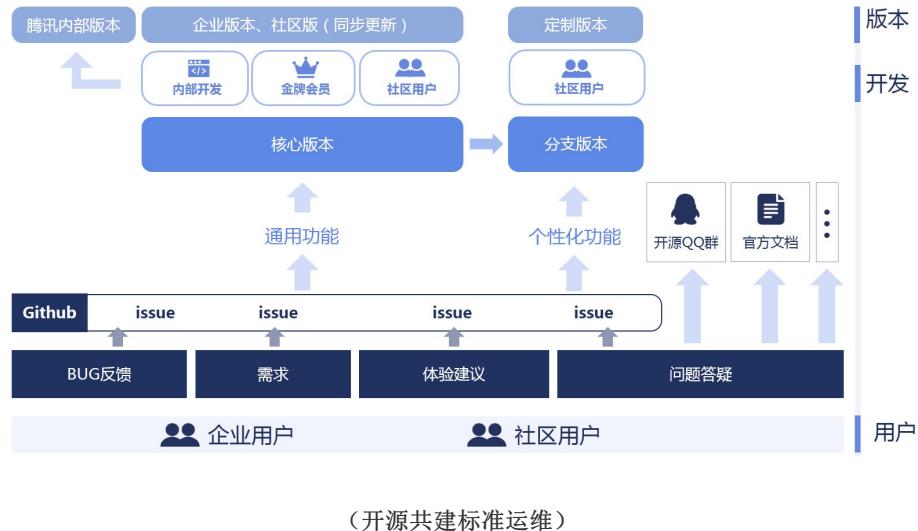
现在你还在通过二进制的方式部署你的业务，但再过 2 年可能就不是这样了，已有不少行业已在使用容器编排服务，部署模式已经发生了革新的转变。所以在蓝鲸体系内，[蓝鲸容器管理平台](#)将补上这块能力的空缺。

万丈高楼平地起，标准化、自动化需要依次建设，技术没有弯道超车。

最后希望，企业的 IT 系统建设能够逐步演进、完善，让广大运维兄弟真正能交付运维价值，体会运

维这个岗位给企业带来的价值。

来吧，一起共建属于大家的开源自动化运维项目：**标准运维！**



## 扩展阅读

- [1] 阮一峰. 持续集成是什么? [EB/OL], 2015-9-23
- [2] 唐盛军. 全球公有云编排服务大比拼 [EB/OL], 2018-6-4
- [3] 中国信息通信研究院. 《混合云白皮书（2019 年）》 [EB/OL], 2019-6-17
- [4] Terraform. Terraform vs. CloudFormation, Heat, etc. [EB/OL]
- [5] 叶光芳. 赣州银行利用蓝鲸标准运维实现容灾一键切换 [EB/OL], 2019-5-28
- [6] 元鼎科技. 金融行业落地自动化持续交付 [EB/OL], 2019-4-9
- [7] 蓝鲸. 企业运维自动化新玩法 [EB/OL], 2019-3-14
- [8] 蓝鲸. Github Tencent/bk-sops [EB/OL]
- [9] 蓝鲸. Github Tencent/bk-cmdb [EB/OL]
- [10] 蓝鲸. Github Tencent/bk-ci [EB/OL]
- [11] 蓝鲸. Github Tencent/bcs [EB/OL]

# 管控混合云架构下的基础设施

## 情景

随着云计算浪潮的推进，多云管控逐渐成为趋势，多云间网络无法互通。

接下来看下蓝鲸是如何管控多云主机。

## 前提条件

- 部署完蓝鲸

## 术语解释

- GSE Server**: 蓝鲸管控平台的后台服务，由文件分发、命令执行、数据上报三个模块组成。
- GSE Proxy**: 跨云网络中，GSE Server 和跨云网络中被管控主机的代理，实现对跨云网络主机的管控。
- Agent**: 运行在被管控主机上的代理程序，实现文件分发、命令执行以及数据上报。
- VPC** : (Virtual Private Cloud)，私有网络，逻辑隔离的网络空间，每个私有网络内的服务资源内网互通，不同私有网络之间内网不通，但可通过新建对等连接来连通。
- 云区域** : 标识 VPC 网络，蓝鲸多云管控的关键字段，通过租户 ID( `bk_supplier_id` )、云区域、IP 三者唯一标识主机。
- 直连网络** : 蓝鲸后台服务所在的网络，该网络下管控的主机与蓝鲸后台互通。

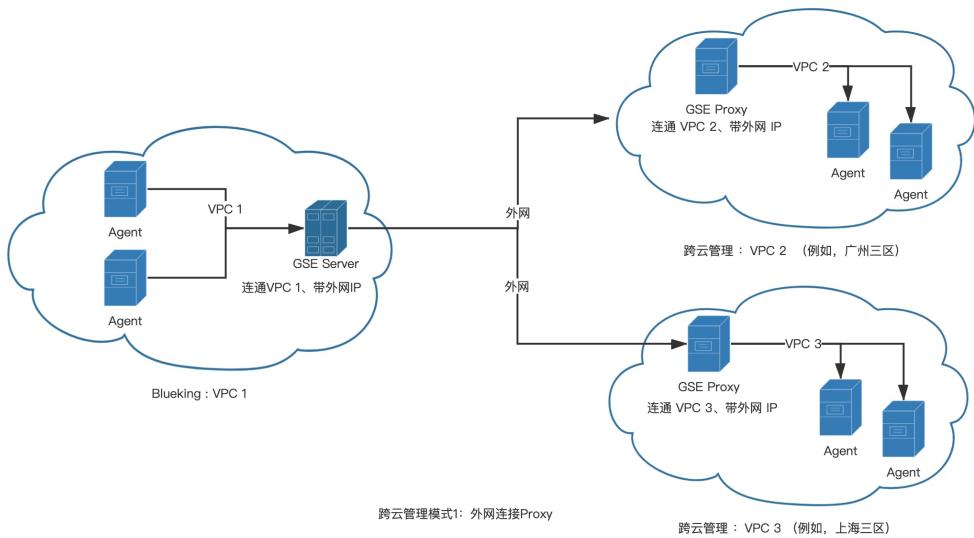
## 操作步骤

- 梳理网络拓扑
- 管理直连网络区域的主机
- 管理跨云网络区域的主机

### 梳理网络拓扑

以下是一个经典的多云管控网络拓扑图。

蓝鲸所在的网络为 **VPC 1**，通过**直连方式**管理该网络区域的主机，通过带有外网的 **GSE Proxy** 来管理公有云的主机（网络为 **VPC 2** 和 **VPC 3**）。



“

上图是从管控角度绘制的网络拓扑图。

”

在跨云管控的场景下，在蓝鲸所在 VPC 1 网络下发安装 Agent 的行为，还需要蓝鲸后台的 **Nginx 模块** 具备外网 IP，供 VPC 2 和 VPC 3 网络的 GSE Proxy 下载 Proxy 和 Agent 安装包。

## 管理直连网络区域的主机

先介绍如何管控蓝鲸后台服务所在网络（也称直连网络）的主机。

### 安装 AGENT

打开 **节点管理**，选择 **直连区域**，按提示填写 **IP地址**、**操作系统**、**端口**、**账号**、**密码**。

节点类型	IP地址	操作系统	端口	账号	校验方式	操作
Agent	10.0.4.29	Linux	SSH	22	root	密码验证 <input checked="" type="checkbox"/> ***** <a href="#">添加</a> <a href="#">删除</a>

Agent 服务器需要满足以下条件

1. APPO 所在机器(在 app 运行所在机器)必须能通过 ssh 登陆到 Agent 机器
2. Agent 所在机器可以访问到 zk 的端口
3. 支持 Linux/Windows/AIX 操作系统
4. Windows 服务器如果没有安装 Cygwin，则需要开通SMB服务(网上邻居)的445端口
5. 必须使用 root/Administrator 账户

[高级设置](#) [点击开始安装](#) [仅导入机器](#)

点击 **开始安装**，稍等片刻安装完成。

The screenshot shows the 'Node Management' interface. At the top, there are tabs for '所有区域' (All Regions), '直连区域' (Direct Connect Region) (selected), '腾讯云' (Tencent Cloud), and '云区域管理' (Cloud Region Management). Below the tabs, there are buttons for '+ 添加Agent' (Add Agent), '从CMDB导入' (Import from CMDB), and '批量导入' (Batch Import). On the right, there are buttons for '升级' (Upgrade), '重装' (Reinstall), '卸载' (Uninstall), '删除' (Delete), '编辑' (Edit), and '终止' (Terminate). A table lists one node: IP地址 (IP Address): 10.0.4.29, 节点类型 (Node Type): Agent, 状态 (Status): 1.60.54, 更新时间 (Last Update): 2019-07-30 18:36:31, 安装步骤信息 (Installation Step Information): 任务执行成功(安装) (Task execution successful (installation)).

在 **插件管理** 页面可以查看该主机上运行的插件状态和版本号。

The screenshot shows the 'Plugin Management' interface. It includes three steps: 1. Select host, 2. Select operation type, and 3. View details. The host selection table lists nodes by IP address, operating system, and installed plugins (gseagent, basereport, unifylogic, processbeat, bkmetricbeat). The row for IP 10.0.4.29 is highlighted with a red border. The bottom navigation bar shows page 1.

在配置平台的 **主机详情页**，可以看到该台主机的 **实时状态**。

The screenshot shows the 'Host Detail Page' in the configuration platform. The left sidebar has categories like 首页 (Home), 基础资源 (Basic Resources), 业务资源 (Business Resources) (selected), 业务主机 (Business Host) (selected), 业务拓扑 (Business Topology), 进程管理 (Process Management), 动态分组 (Dynamic Group), 审计与分析 (Audit and Analysis), 权限控制 (Permission Control), 模型管理 (Model Management), and 我的收藏 (My Favorites). The main area shows a host detail for IP 10.0.4.29. It includes tabs for 属性 (Properties), 关联 (Associations), 实时状态 (Real-time Status) (selected), and 变更记录 (Change Log). Real-time status metrics include: 总流入量: 0.00Mb/s, 总流出量: 0.00Mb/s, 内存总量: 0.97GB, 最近更新时间: 2019-07-30 18:39:10, 启动时间: 2019-07-30 10:47:33, 磁盘总量: 49GB, loadavg: 0.03 0.04 0.05. Below these are three gauge charts: 总CPU使用率 (Total CPU Usage Rate) at 2.68%, 总内存使用率 (Total Memory Usage Rate) at 36.31%, and 磁盘使用情况 (Disk Usage Situation) at 10.2%.

在 **主机详情页** 的属性 TAB 页，可以看到 **自动发现** 的主机属性，源于运行在被管控主机上的 **basereport** 插件主动上报。

在 **变更记录** TAB 页可以知道，节点管理在安装 Agent 的过程中会向配置平台导入主机。

## 命令执行和分发文件测试

使用作业平台 **执行脚本** 和 **分发文件** 做测试。

## 管理跨云网络区域的主机

接下来介绍，如何管控跨云网络（例如 VPC 2 和 VPC 3）的主机。

## 策略开通

跨云网络中，通过**GSE Proxy**代理 GSE Server 与跨云网络中被管控主机的流量，实现对跨云网络主机的管控。

点击节点管理右侧的【?帮助】菜单，会显示节点管理安装 Agent、Proxy 的架构图，以及 GSE Proxy 与 GSE Server 通信相互开通的策略。



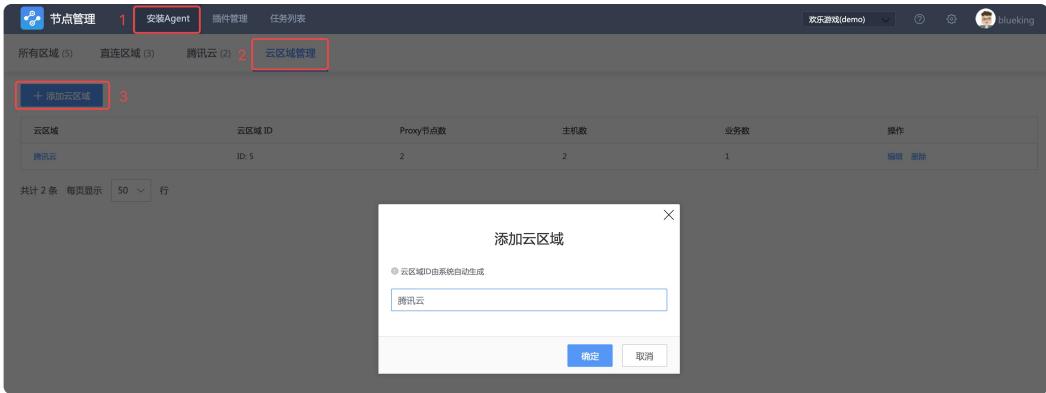
完成上述网络访问策略开通,接下来，准备安装 GSE Proxy。

## 安装 GSE PROXY

为了演示方便，将网络拓扑图中的 VPC 2 命名为“腾讯云”，你也可以命名为“广州三区”或“上海三区”等。

现在添加云区域，如 **腾讯云**。

选择菜单【安装 Agent】，点击【云区域管理】，然后【添加云区域】，如“腾讯云”。



填写 Proxy 的内网、外网 IP 地址，以及账号，点击【开始安装】。



完成 Proxy 的安装，在【云区域管理】列表中可以找到已经安装好的 Proxy。



完成 Proxy 的安装，接下来安装该网络下的 Agent。

## 安装 AGENT

选择菜单【安装 Agent】，点击【腾讯云】，【添加 P-Agent】，然后点击【开始安装】。

节点类型 IP地址 操作系统 端口 账号 校验方式 操作

P-Agent	10.0.15.180	Linux	SSH 22	root	密码验证	*****	添加	删除
P-Agent	10.0.15.62	Linux	SSH 22	root	密码验证	*****	添加	删除

安装 P-Agent 需要满足以下条件  
 1. 支持 Linux/Windows/AD 操作系统  
 2. Linux/AD 可以从任意 Proxy 节点通过 ssh 登陆到目标 P-Agent 节点  
 3. Windows 服务器需要开放 139、445 端口，可以让该区域内的任意 Proxy 节点访问  
 4. 开通 P-Agent 与 Proxy 之间的网络策略（参考 mini 文档中的网络策略说明）  
 5. 必须使用 root/Administrator 账户

高级添加 点击开始安装 仅导入机器

完成该网络下的 Agent 安装。

IP地址	节点类型	操作系统	Agent状态	Agent版本	更新时间	任务状态	操作
10.0.15.180	P-Agent	Linux	正常	1.60.54	2019-08-02 08:25:01	任务执行成功(重装)	编辑 升级 更多
10.0.15.62	P-Agent	Linux	正常	1.60.54	2019-08-02 08:25:01	任务执行成功(重装)	编辑 升级 更多

共计 1 页 每页显示 50 行

在【配置平台】中也可以找到对应的主机和主机属性。

业务拓扑

10.0.15.62

属性 关联 实时状态 变更记录

业务拓扑 欢乐游戏(demo) > 空闲机 > 空闲机

基础信息

交换机端口: --	内网IP: 10.0.15.62
外网IP: --	固件编号: --
主要维护人: --	备份维护人: --
服务器类型: 物理机	设备SN: --
备注: --	质保年限: --
SLA级别: --	云区域: 腾讯云
所在国家: --	所在省份: --
所属机架号: --	存放机架ID: --
连接设备ID: --	

自动发现信息 (需要安装agent)

主机名称: VM_15_02_centos	操作系统类型: Linux
操作系统名称: Linux centos	操作系统版本: 7.6.1810
操作系统位数: 64-bit	CPU逻辑核心数: 1
CPU频率: 1999 Hz	CPU型号: AMD EPYC Processor
内存容量: 991 MB	磁盘容量: 49 GB
内网MAC地址: 52:54:00:64:15:d4	外网MAC: --

更多属性 编辑

## 命令执行和分发文件测试

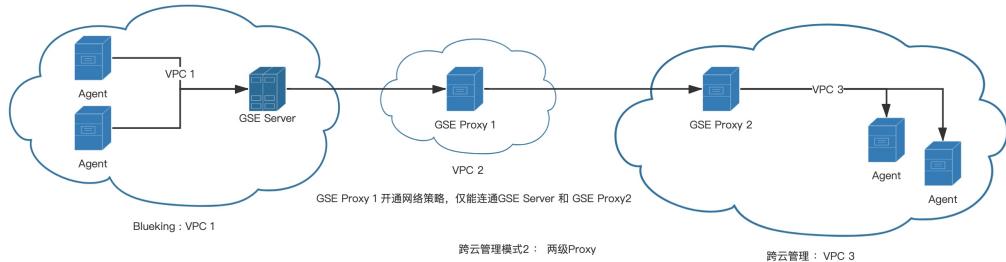
使用作业平台 **执行脚本** 和 **分发文件** 做测试。

```
{% video %}media/bk_nodeman.mp4{% endvideo %}
```

## 扩展阅读

## 多级级联：管理隔离网络的主机

在部分企业网络环境中，存在 **VPC 1**（蓝鲸所处网络） 和 **VPC 3** 不互通，但需要管控 VPC 3 网络主机的场景，如下图：



这时候，需要企业网络环境中，存在一个 VPC 2 的网络，通过申请网络策略，连通 VPC1 和 VPC 3 中的指定（最小策略，安全）服务器。

这时需要使用蓝鲸管控平台的多级级联功能，VPC 3 网络中的主机与 GSE Proxy 2 互通，GSE Proxy 2 通过上联 GSE Proxy 1 与 蓝鲸 GSE Server 通信。

“

多级级联 功能尽请期待。

”

# 37秒完成万台服务器的目录标准化

## 情景

对运行在上万台服务器上的业务服务做标准化的调整（还历史债务），经过多轮的灰度验证，计划对剩下数万台服务器批量操作。

## 前提条件

- 服务器已在 **CMDB 注册**
- 拥有服务器所在 CMDB 中业务的运维权限

## 操作步骤

- 新建作业
- 执行和查看执行结果

## 新建作业

按照标准化的需求，我们需要将 `gsectl` 文件推送至 `/usr/local/gse_bkte/agent/bin/` 目录，为了确保万无一失，做 MD5 校验。

作业模板如下：

The screenshot shows the Blue King Job Platform interface. It displays two job templates:

- 标准gsectl推送**: This template is for pushing the `gsectl` file. The target path is set to `/usr/local/gse_bkte/agent/bin/`. The file list shows a single file: `1./data/gse_publish/agent/gsectl/gsectl`.
- gsectl md5版本检查**: This template is for checking the MD5 version. It has a script name of `gsectl md5版本检查` and a script source type of "脚本引用".

提示：为了阐述作业平台中的两个全局参数：`IP` 和 `云参`，我们通过 **需求自助化** 中用到的作业模板来介绍。

The screenshot shows the Blue King Job Platform interface with the following sections:

- 全局IP设置**: This section allows setting global IP parameters. It includes fields for "选择服务器" (selected: `job_host`), "分组名" (selected: `自助调整角色`), and "选择需要替换的节点IP" (status: 1台正常 0台异常). Buttons include "全局变量设置" (highlighted with a red box), "复制IP", "复制Agent未安装IP", "清空IP", and "清空Agent未安装". A note at the bottom says "共1台: 0个静态IP, 1个分组, 正常: 1, 未安装: 0".
- 云参**: This section allows defining cloud parameters. A table lists a parameter named `role` with a value of `角色`.

`IP` 这个参数，我们一般建议使用 **动态分组**，因为服务器会有故障替换的可能，`IP` 会变。

`云参` 在脚本中可以直接引用。

步骤名称：自助调整角色

脚本名称 \* 服务器账户 \* 服务器数 \* 脚本参数 所有操作

脚本全局变量替代 IP已经被全局变量替代  敏感参数  完成后暂停  ↑ ↓ ×

脚本来源 \*: 手工录入 脚本引用 脚本克隆 本地脚本 公共脚本

脚本名称: 自助调整角色 版本号: blueking.2019072311340... 版本备注: V1.0 保存为新版本

脚本内容 \*:

```

20 ##### 可在脚本执行失败的逻辑分支处调用，打印当时的返回值及PID。
21 function job_fail
22 {
23     MSG="$"
24     echo "$MSG` eval \$NOW` job_fail:[\$MSG]"
25     exit 1
26 }
27
28 job_start
29
30 ##### 可在此处开始编写您的脚本逻辑代码
31 ##### 作业平台中执行脚本成功和失败的标准只取决于脚本最后一条执行语句的返回值
32 ##### 如果返回值为0，则认为此脚本执行成功，如果非0，则认为脚本执行失败
33
34 echo $role
35
36
37
38

```

超时时间： 1000

目标机器 \*: 已被全局变量替代, 点击查看!

## 执行作业及查看执行结果

点击 **执行作业** 后，在 9946 台服务器上总耗时 37 秒。

当前业务：  帮助中心

作业执行 > 执行详情 作业实例

作业名称：[作业] 标准gsecl推送 执行结果： 执行成功 启动人：  总耗时(s)： 37.02

作业步骤

步骤名称：[作业] 标准gsecl推送							
序号	任务名称	执行主机数	开始时间	结束时间	总时间(s)	状态	所有操作
1	脚本标准gsecl推送	9946	2019-07-18 18:25:39	2019-07-18 18:25:52	12.934	执行成功	<input type="button" value="执行详情"/>

步骤名称：gsecl md5版本检查							
序号	脚本名称	执行主机数	开始时间	结束时间	总时间(s)	状态	所有操作
1	gsecl md5版本检查	9946	2019-07-18 18:26:53	2019-07-18 18:26:17	24.087	执行成功	<input type="button" value="执行详情"/>

其中分发 gsecl 文件耗时 13 秒。

节点名称	节点状态	节点类型	开始时间	服务器租户	总时间(s)
gseccli推送	执行成功	文件传输	2019-07-18 18:25:39	gse	12.934

完整日志  
[2019-07-18 18:25:41.26] [Upload source files] FileName : /data/gse\_publish/agent/gseccli/gseccli FileSize : 7.621 KB State : upload Speed : 7KB/s Progress : 100% Detail : Success  
[2019-07-18 18:25:41.26] FileName : /usr/local/gse\_bkte/agent/bin/gseccli FileSize : 7.621 KB State : download Speed : 7KB/s Progress : 100% Detail : succ

MD5 校验耗时 24 秒。

节点名称	节点状态	执行脚本	开始时间	服务器租户	总时间(s)
gseccli md5版本检查	执行成功	job_start	2019-07-18 18:25:53	gse	24.087

完整日志  
[2019-07-18 18:25:54][PID:14629] job\_start  
[2019-07-18 18:25:54][PID:14629] job\_success:[gseccli md5sum OK]

## 一次标准的应用交付自动化案例

### 情景

应用发布是运维这个岗位的职能之一，发布关联多个 ITIL 系统的功能模块，比如发布单、监控的告警屏蔽、DB 变更、业务内公告、统一登录入口等，频繁在多个系统间切换，不但影响效率而且容易出错，同时无法可视化查看发布进度以及事后的回溯。

接下来，一起看下标准运维是如何解决这些痛点。

## 前提条件

- 主机在蓝鲸 CMDB 中管理

## 术语解释

- 流程模板**：标准化的资源和应用交付模式，通过资源编排引擎，实现对资源的创建、配置，实现自动化交付资源或应用，行业中一般称之为 **pipeline**、**资源编排模板**，比如一次发布任务可以编排为一个流程模板。
- 标准插件**：多个执行节点通过编排规则实现流程模板，其中的执行节点称之为 **标准插件**，比如 **执行脚本** 为一个标准插件

更多详见 [术语定义](#)

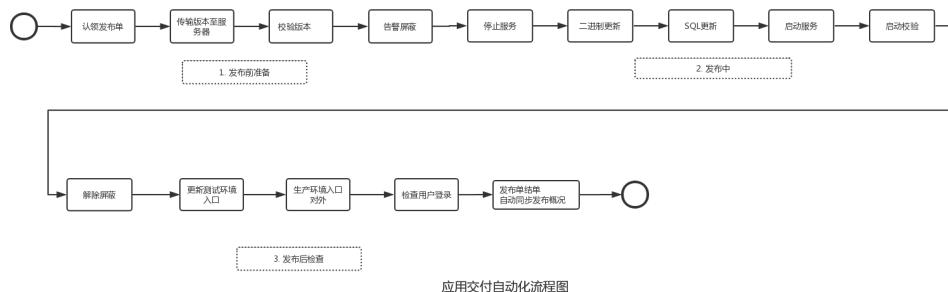
## 操作步骤

- 梳理：梳理标准化模板
- 建模：新建流程模板
- 执行：执行流程

### 梳理标准化模板

[配置管理标准化](#) 中提到，运维服务“四化建设”的标准化包含配置管理、发布、变更、故障处理、监控告警等场景的流程制定。以发布为例，通过流程图梳理应用交付的流程。

分为发布前准备、发布中、发布后检查三部分。

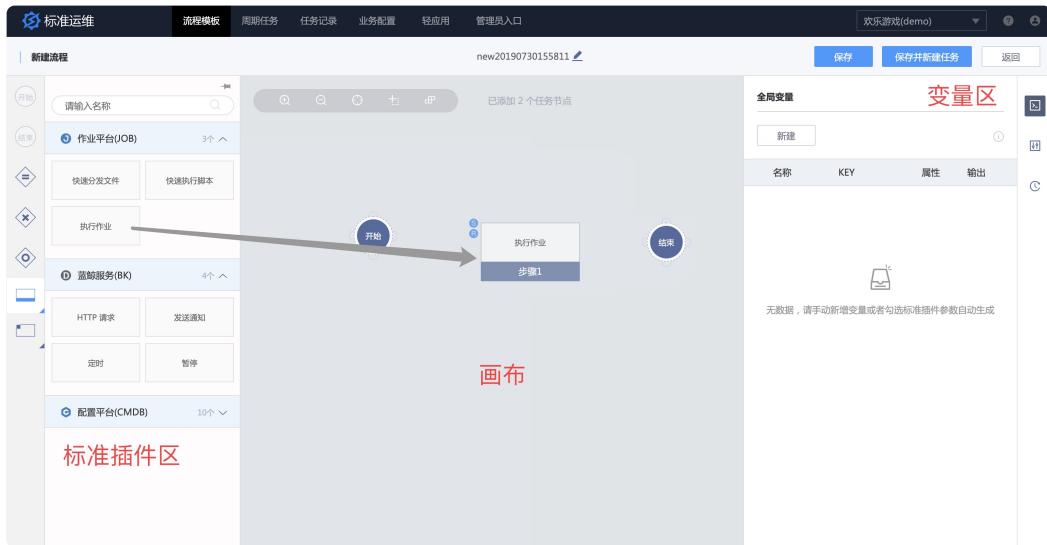


## 创建流程模板

为了简化演示，将流程图中的关键节点在标准运维的业务流程模板中体验出来。

选择 [流程模板] -> [业务流程]，点击 新建 来创建业务流程模板。

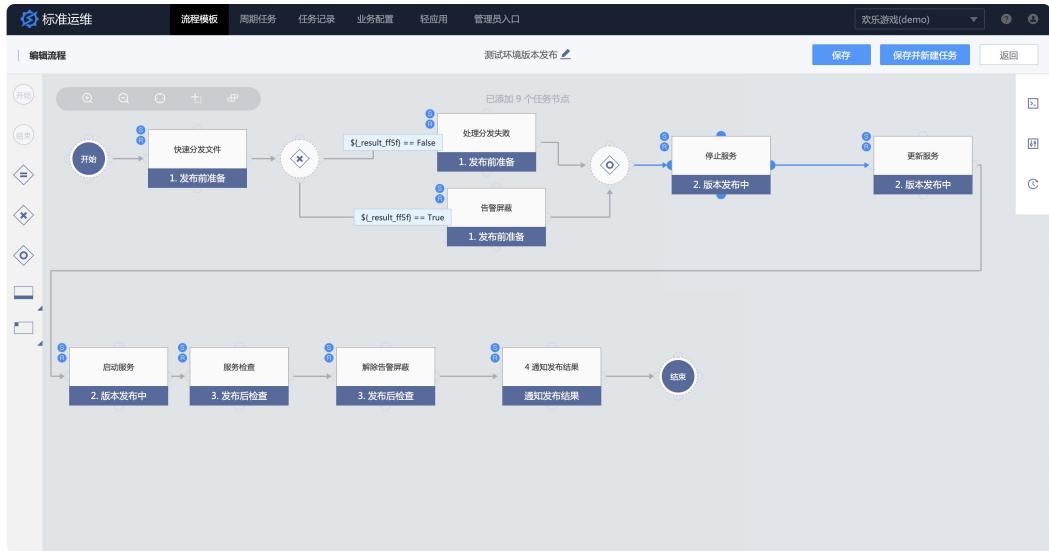
从左侧标准插件区，选择发布流程中需要的标准插件作为流程的节点，比如 执行作业，向右拖动到画布。



选择作业平台中准备好的 作业模板，然后新建 全局变量，并将全局变量填充到节点的参数中。



按照上述步骤，完成一个应用发布的流程模板。



“ 标准运维内置了 CMDB、作业平台、通知等标准组件，如果不在此列的，需要开发标准运维插件来 **集成企业内部 ITIL 系统**。 ”

这里重点说明 **全局参数** 和 **流程分支**。

## 全局参数

服务器发生故障后，保障下一次应用发布获取最新的 IP 列表，可以通过 IP 选择器实现。

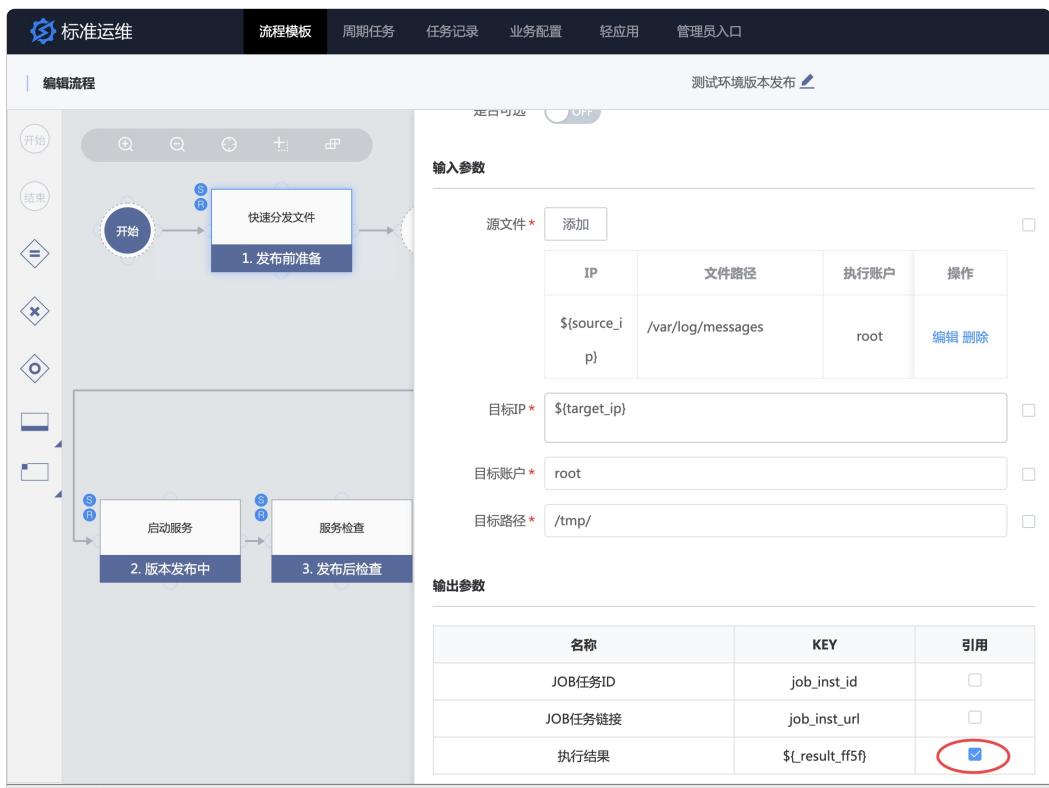


## 流程分支

应用发布过程中，执行成功 和 执行失败的处理分支不同，可以通过流程分支功能对上一步执行结果为真或为假来判断。



提前引用上一步流程节点的输出参数 **执行结果**，将其用于上图中的流程分支表达式。



## 执行流程

在业务流程列表中，点击 **新建任务**

点击执行任务流程

{% video %}media/sops\_execution.mp4{% endvideo %}

## 扩展阅读

### 上下文传参

将一个流程节点的输出作为另一个流程节点的输入。

比如第 1 步输出 MD5 值，第 2 步分发版本，第 3 步使用第 1 步中生成的 MD5 值来校验版本的一致性，效果如下：

主要用到标准运维流程节点中的 引用输出参数，引用第 1 步中的 `release_md5` 变量。



`release_md5` 变量需要提前在作业模板中设置，如下图：

The screenshot shows the 'Job Template' configuration interface. It includes sections for '作业属性' (Job Properties), '全局IP设定' (Global IP Settings), '云参' (Cloud Parameters), and '脚本内容' (Script Content). The '脚本内容' section displays a shell script with a line highlighted: `release_md5=$(md5sum /var/log/messages | awk '{print $1}'')`. A note at the bottom right states: '已被全局变量替代, 点击查看!' (Replaced by global variable, click to view!).

## 操作员做应用交付，让运维更专注

### 情景

应用交付虽然做到了自动化，但运维仍需和周边团队跟进流程，耗费大量时间，此时通过 **标准运维职能化** 的能力将应用交付的流程，转移给操作员执行。

运维 **blueking** 提交一个应用发布的职能化任务给到 **caozuo(操作员A)**，**caozuo(操作员A)** 执行任务，跟进任务执行的进展。

## 前提条件

- 在标准运维中 **建立一个流程**
- 准备两个角色：**运维** 和 **职能化(操作员)**

## 操作步骤

- 准备职能化角色
- 运维发起职能化任务流程
- 职能化认领任务

### 准备职能化角色

在 **用户管理** 中新增 **职能化账户**，如 **操作员A**

The screenshot shows the 'User Management' page in the BlueKing platform. At the top, there are buttons for 'Add User' (+), 'Batch Import' (批量导入), and 'Export' (导出). Below the search bar, a table lists users. A new user row for '操作员A' is being added, with its 'Role' dropdown set to 'Operational User'. The 'Save' (保存) button is highlighted with a red box.

将其添加至业务的 **操作人员** 角色中

The screenshot shows the 'Business' section of the BlueKing Configuration Platform. On the left sidebar, 'Business' is selected. In the main area, a service named '欢乐游戏(demo)' is shown. Under the 'Roles' tab, the '操作人员' role is assigned to the user 'caozuo(操作员A)', which is also highlighted with a red box.

在标准运维的 **业务流程** 中给 **职能化** 授权

The screenshot shows the 'Business Process' section of the Standard Operation interface. A modal window titled '权限管理' (Permission Management) is displayed, listing three rows of permissions assigned to users 'blueking' and 'breaking'. The '操作' (Operation) column for user 'breaking' is highlighted with a red circle.

## 运维发起职能化任务流程

运维选择一个应用发布流程，点击 **新建任务**，选择 **职能化任务流程**

The screenshot shows the 'New Task' creation process in the Standard Operation interface. In the 'Task Type' section, 'Default Task Flow' is selected, while 'Functional Task Flow' is highlighted with a red box.

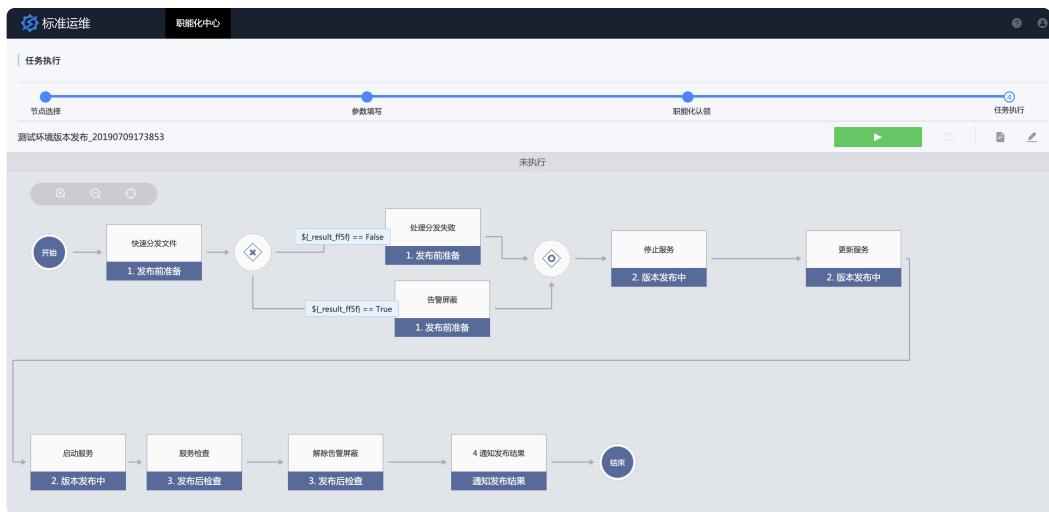
## 职能化认领任务

用 **操作员A** 的角色访问标准运维，在 **职能化中心** 中认领刚才新建的 **职能化任务**

The screenshot shows the 'Functionalization Center' tasks list in the Standard Operation interface. A task for 'Test Environment Version Release' is listed, with its ID highlighted in a red box. The '认领' (Claim) button for this task is also highlighted in a red box.

如果不需要修改参数，点击 **认领** 即可

点击 **执行** 按钮



可以看到任务执行的状态 { % video %}media/stag\_delivery.mp4{ % endvideo %}

在 **职能化中心** 可以看到任务已经执行完成。

The screenshot shows a table of tasks under the '任务记录' tab. The columns include: 所属业务 (Business Unit), 任务ID (Task ID), 任务名称 (Task Name), 提单时间 (Submitted Time), 认领时间 (Accepted Time), 提单人 (Submitter), 认领人 (Assignee), 状态 (Status), and 操作 (Operations). The tasks listed are all completed ('完成') by 'blueking'.

所属业务	任务ID	任务名称	提单时间	认领时间	提单人	认领人	状态	操作
欢乐游戏(demo)	37	测试环境版本发布_20190709200241	2019-07-09 20:04:54 +0800	2019-07-09 20:06:46 +0800	blueking	caozuo	● 完成	查看
欢乐游戏(demo)	36	测试环境版本发布_20190709159512	2019-07-09 20:01:18 +0800	2019-07-09 20:01:35 +0800	blueking	caozuo	● 已执行	查看
欢乐游戏(demo)	35	测试环境版本发布_20190709173853	2019-07-09 17:41:06 +0800	2019-07-09 17:48:47 +0800	blueking	caozuo	● 已执行	查看
欢乐游戏(demo)	34	测试环境版本发布_20190701162612	2019-07-01 16:45:42 +0800	--	zhinenghua	--	● 未认领	认领
欢乐游戏(demo)	33	测试环境版本发布_20190701160313	2019-07-01 16:05:06 +0800	2019-07-01 16:05:12 +0800	zhinenghua	zhinenghua	● 已执行	查看
欢乐游戏(demo)	32	测试环境版本发布_20190701160154	2019-07-01 16:04:02 +0800	2019-07-01 16:04:27 +0800	blueking	zhinenghua	● 已执行	查看

对应运维视角，也可以看到任务已执行完成。

The screenshot shows a table of tasks under the '任务记录' tab. The columns include: ID, 任务名称 (Task Name), 执行开始 (Start Time), 执行结束 (End Time), 任务分类 (Task Category), 创建人 (Creator), 执行人 (Executor), 创建方式 (Creation Method), 状态 (Status), and 操作 (Operations). The tasks listed are all completed ('完成') by 'blueking'.

ID	任务名称	执行开始	执行结束	任务分类	创建人	执行人	创建方式	状态	操作
37	测试环境版本发布_20190709200241	2019-07-09 20:08:09 +0800	2019-07-09 20:08:31 +0800	运维工具	blueking	caozuo	手动	● 完成	克隆 删除
36	测试环境版本发布_20190709159512	2019-07-09 20:01:47 +0800	--	运维工具	blueking	caozuo	手动	● 失败	克隆 删除
35	测试环境版本发布_20190709173853	2019-07-09 19:46:15 +0800	--	运维工具	blueking	caozuo	手动	● 失败	克隆 删除
34	测试环境版本发布_20190701162612	--	--	运维工具	zhinenghua	--	手动	● 未执行	克隆 删除
33	测试环境版本发布_20190701160313	2019-07-01 16:05:17 +0800	--	运维工具	zhinenghua	zhinenghua	手动	● 失败	克隆 删除
32	测试环境版本发布_20190701160154	2019-07-01 16:04:34 +0800	--	运维工具	blueking	zhinenghua	手动	● 失败	克隆 删除

如此，运维可以把日常的应用发布需求提交给操作员执行，释放自己，把精力多放在 [运维服务目录输出](#) 以及 [业务优化](#) 上。

## 扩展阅读：设置任务执行者

操作员执行任务流程时，标准运维从 [任务执行者](#) 字段获取运维人员列表，如果不填，则为该业务的随机运维人员。

The screenshot shows the 'Executor Configuration' section. It includes a text input field for '任务执行者' (Executor) with a placeholder '该字段默认在非运维人员执行任务时生效，为空则从配置平台(CMDB)随机获取运维身份' (This field is default to be effective when non-maintenance personnel execute tasks, and it will be randomly selected from the configuration platform (CMDB)). Below the input field is a checkbox labeled '强制生效' (Force Activation) with the value 'OFF'. At the bottom right is a green '保存' (Save) button.

# 开发标准运维插件，集成内部 IT 系统

## 情景

标准运维内置了蓝鲸体系内作业平台、配置平台等系统的原子，但应用交付过程中还包含了部分企业内部的 ITIL 系统，例如 [DB变更](#)、[监控](#)、[工单](#)，需要开发标准运维的标准插件，将应用交付过程

中，使用到的能力或接口集成到标准运维中。

## 前提条件

- 掌握 蓝鲸 SaaS 开发，打开 [腾讯运维开发实战课](#) 马上学习
- 掌握 蓝鲸 API 网关开发

## 步骤

- 梳理逻辑
- 开发环境初始化
- 蓝鲸 API 网关开发
- 标准插件后台开发
- 标准插件前端开发
- 标准插件测试

### 梳理逻辑

标准运维要调用 IT 系统的功能特性，比如 [执行 DB 变更](#) 或 [告警屏蔽](#)，需要将对应 API 对接至蓝鲸 ESB 中，然后再开发标准运维的标准插件。

以下为标准插件的调用逻辑图。

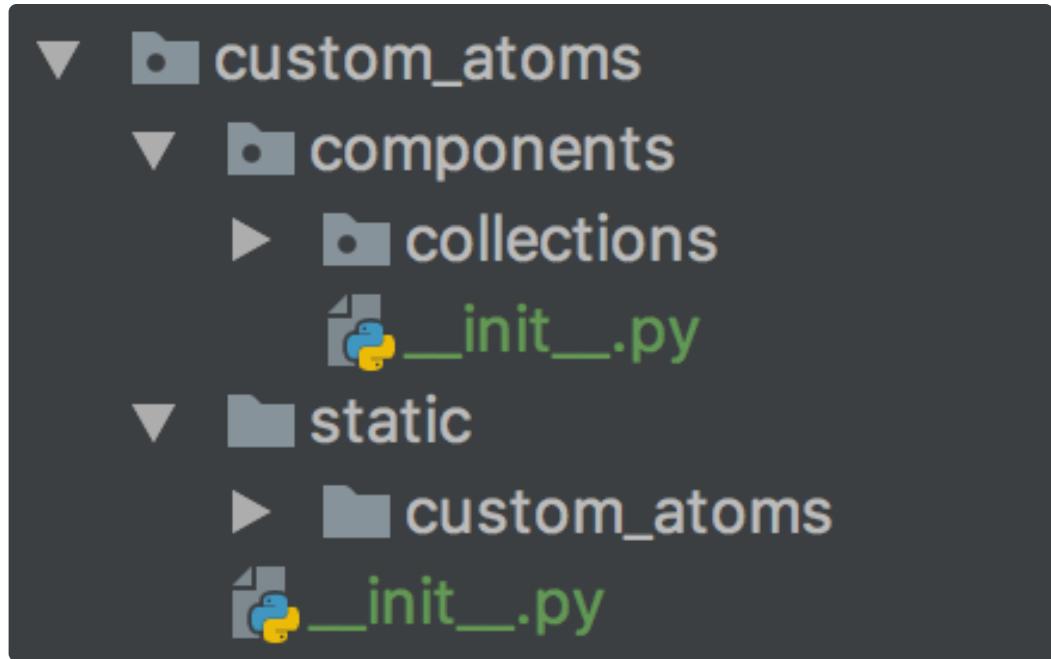


注：建议 IT 系统的功能特性和标准运维解耦，标准运维不包含功能逻辑，仅负责转发。

### 开发环境初始化

在开始开发之前，先把 [蓝鲸 SaaS 的开发环境](#)准备好了。

然后在 标准运维项目根目录 下执行 `Django-admin startapp custom_atoms`，接着新建 `components/collections` 和 `static/custom_atoms` 目录。



打开 `conf/settings_custom.py` 文件，找到 `INSTALLED_APPS_CUSTOM`，加入 `custom_atoms`。

```
INSTALLED_APPS_CUSTOM = (
    # add your app here...
    'custom_atoms'
)
```

## 接入 ESB API

参照 蓝鲸 API 网关开发指南完成 ESB 接入，然后更新标准运维 `blueking/component` 下的文件。

The screenshot shows a file tree on the left and a code editor on the right. The file tree under 'blueking/component/apis' includes files like \_\_init\_\_.py, bk\_login.py, cc.py, cmsi.py, gse.py, job.py, test\_api.py, \_\_init\_\_.py, base.py, client.py, collections.py, conf.py, exceptions.py, README, shortcuts.py, and utils.py. The code editor displays 'collections.py' with the following content:

```
# -*- coding: utf-8 -*-
"""Collections for component client"""
from .apis.bk_login import CollectionsBkLogin
from .apis.cc import CollectionsCC
from .apis.cmsi import CollectionsCMSI
from .apis.job import CollectionsJOB
from .apis.gse import CollectionsGSE
from .apis.test_api import CollectionsTEST

# Available components
AVAILABLE_COLLECTIONS = {
    'bk_login': CollectionsBkLogin,
    'cc': CollectionsCC,
    'cmsi': CollectionsCMSI,
    'job': CollectionsJOB,
    'gse': CollectionsGSE,
    'test_api': CollectionsTEST,
}
```

Two red arrows point to the line 'from .apis.test\_api import CollectionsTEST' and the entry 'test\_api': CollectionsTEST, in the AVAILABLE\_COLLECTIONS dictionary.

## 标准插件后台开发

在 `custom_atoms/components/collections` 目录下创建 `test.py` 文件，其中需要定义的属性和类如下所示。

```

# -*- coding: utf-8 -*-
from blueapps.utils.esbclient import get_client_by_user

from pipeline.conf import settings
from pipeline.core.flow.activity import Service
from pipeline.component_framework.component import Component

__group_name__ = u"测试原子(TEST)"


class TestCustomService(Service):

    def execute(self, data, parent_data):
        executor = parent_data.get_one_of_inputs('executor')
        biz_cc_id = parent_data.get_one_of_inputs('biz_cc_id')
        client = get_client_by_user(executor)

        test_input = data.get_one_of_inputs('test_input')
        test_textarea = data.get_one_of_inputs('test_textarea')
        test_radio = data.get_one_of_inputs('test_radio')

        api_kwargs = {
            'bk_biz_id': biz_cc_id,
            'executor': executor,
            'test_input': test_input,
            'test_textarea': test_textarea,
            'test_radio': test_radio,
        }
        api_result = client.test_api.test1(api_kwargs)

        if api_result['result']:
            data.set_outputs('data', api_result['data'])
            return True
        else:
            data.set_outputs('ex_data', api_result['message'])
            return False

    def outputs_format(self):
        return []


class TestCustomComponent(Component):
    name = u"自定义测试"
    code = 'test_custom'
    bound_service = TestCustomService
    form = settings.STATIC_URL + 'custom_atoms/test/test_custom.js'

```

**test.py 属性详解:**

- `__group_name__` : 标准插件所属分类 (一般是对应 API 的系统简称, 如配置平台(CMDB))
- `class TestCustomService(Service)` : 标准插件后台执行逻辑
- `__need_schedule__ = True`: 是否是异步执行, 默认为 False
- `interval = StaticIntervalGenerator(5)` : 异步标准插件的轮询策略

- `def execute` : 前端参数获取、API 参数组装、结果解析、结果输出
- `def schedule` : 轮询逻辑、结果输出
- `def outputs_format` : 输出参数格式化
- `class TestCustomComponent(Component)` : 标准插件定义
- `name` : 标准插件名称
- `code` : 唯一编码
- `bound_service` : 绑定后台服务
- `form` : 前端表单定义文件路径

**TestCustomService** 中 `execute` 函数详解: - 可以是任何 Python 代码, 如果对应于 ESB API 调用, 一般分为参数组装、API 调用、结果解析。- `data` 是标准插件前端数据, 对应于前端的表单, 可以用 `get_one_of_inputs` 获取某一个参数; 执行完成可以使用 `set_outputs` 写入返回值和异常信息 (`ex_data`)。- `parent_data` 是任务的公共参数, 包括 executor—执行者, operator—操作员, biz\_cc\_id—所属业务 ID。详细请查看 `gcloud/taskflow3/utils.py`。- 返回 True 表示标准插件执行成功, False 表示执行失败

```

class TestCustomService(Service):

    def execute(self, data, parent_data):
        executor = parent_data.get_one_of_inputs('executor')
        biz_cc_id = parent_data.get_one_of_inputs('biz_cc_id')
        client = get_client_by_user(executor)

        test_input = data.get_one_of_inputs('test_input')
        test_textarea = data.get_one_of_inputs('test_textarea')
        test_radio = data.get_one_of_inputs('test_radio')

        api_kwargs = {
            'bk_biz_id': biz_cc_id,
            'executor': executor,
            'test_input': test_input,
            'test_textarea': test_textarea,
            'test_radio': test_radio,
        }
        api_result = client.test_api.test1(api_kwargs)

        if api_result['result']:
            data.set_outputs('data', api_result['data'])
            return True
        else:
            data.set_outputs('ex_data', api_result['message'])
            return False
    
```

参数组装

API 调用

结果解析

**TestCustomService** 中 `execute` 函数详解: - 返回列表格式。- 列表格式的每一项定义一个返回字段, 是 `execute` 函数中的 `set_outputs` 输出的字段的子集; key—输出字段标识, name—输出字段含义, type—输出字段类型 (str、int 等 python 数据结构)。

```

def outputs_format(self):
    return [
        self.OutputItem(name=u'JOB任务ID', key='job_inst_id', type='int'),
        self.OutputItem(name=u'JOB任务链接', key='job_inst_url', type='str')
    ]

```

`TestCustomService` 中 `shedule` 函数详解： - 由 `interval` 控制调用策略，如 `pipeline.core.flow.activity.StaticIntervalGenerator`（每隔多少秒轮询一次）、`DefaultIntervalGenerator`（每次轮询间隔时间是上一次的两倍）。 - 使用 `self.finish_schedule` 结束轮询。 - 返回 `True` 表示标准插件执行成功，`False` 表示执行失败。

```

def schedule(self, data, parent_data, callback_data=None):
    task_inst_id = data.get_one_of_outputs('job_inst_id')
    client = data.get_one_of_outputs('client')
    job_kwarg = {
        'task_instance_id': task_inst_id,
    }
    job_result = client.job.get_task_result(job_kwarg)
    if not job_result['result']:
        data.set_outputs('ex_data', job_result['message'])
        self.finish_schedule()
        return False
    # 任务执行结束
    job_data = job_result['data']
    if job_data['isFinished']:
        query = {
            'app': JOB_APP_CODE,
            'url': u'%s?taskInstanceId&appId=%s#taskInstanceId=%s' % (
                settings.BK_JOB_HOST,
                parent_data.get_one_of_inputs('biz_cc_id'),
                task_inst_id,
            )
        }
        job_inst_url = "%s/?%s" % (settings.BK_CONSOLE_URL, urlencode(query))
        # 执行成功
        if job_data['taskInstance'].get('status', 0) in JOB_SUCCESS:
            data.set_outputs('data', job_data)
            data.set_outputs('job_inst_url', job_inst_url)
            self.finish_schedule()
            return True
        # 执行失败
        else:
            data.set_outputs('ex_data', _(u"任务执行失败, <a href='%s' target='_blank'>%s</a>") % job_inst_url,
                            )
    self.finish_schedule()
    return False

```

## 标准插件前端开发

在 `custom_atoms/static/custom_atoms` 目录下创建 `test` 目录，并创建 `test_custom.js` 文件，注意文件路径和标准插件后台定义的 form 保持一致。通过 `$.atoms` 注册标准插件前端配置，其中各项含义是： - `test_custom`：标准插件后台定义的 code。 - `tag_code`：参数 code，请保持全局唯一，命名规范为“系统名\_参数名” - `type`：前端表单类型，可选 input、textarea、radio、checkbox、select、datetime、datatable、upload、combine 等 - `attrs`：对应 type 的属性设置，如 name、validation

(function(){

```
$atoms.test_custom = [
  {
    tag_code: "test_input",
    type: "input",
    attrs: {
      name: "参数1",
      placeholder: "请输入字符串",
      hookable: true,
      validation: [
        {
          type: "required"
        }
      ]
    },
    {
      tag_code: "test_textarea",
      type: "textarea",
      attrs: {
        name: "参数2",
        hookable: true,
        validation: [
          {
            type: "required"
          }
        ]
      }
    },
    {
      tag_code: "test_radio",
      type: "radio",
      attrs: {
        name: "参数3",
        items: [
          {value: "1", name: "选项1"},
          {value: "2", name: "选项2"},
          {value: "3", name: "选项3"}
        ],
        default: "1",
        hookable: true,
        validation: [
          {
            type: "required"
          }
        ]
      }
    }
]
```



## 标准插件测试

创建流程模板，新增标准插件节点，标准插件类型选择新开发的标准插件，展示的输入参数和前端配置项一致，输出参数和后台 `outputs_format` 一致，其中执行结果是系统默认，值是 `True` 或 `False`，表示节点执行结果是成功还是失败。

**基础信息**

标准插件 测试原子(TEST)-自定义测试

节点名称 节点\_1

忽略错误  OFF

是否可选  OFF

**输入参数**

参数1 111

参数2 222

参数3  选项1  选项2  选项3

**输出参数**

名称	KEY	引用
执行结果	_result	<input type="checkbox"/>

>注：勾选[引用]将该参数添加为全局变量！

根据上一步创建的流程模板，新建任务执行后查看结果。

✖ 失败 | 测试新原子\_20180808034939



如果标准插件执行出错，请先查看节点执行详情，确定是否是代码逻辑异常。

节点选择    参数填写    任务执行

失败 | 测试新原子\_20180808034939

### 执行详情

执行信息

开始时间	结束时间
2018-08-08 15:49:43 +0800	2018-08-08 15:49:49 +0800

输入参数

脚本类型 \* 1  
脚本内容 \* exit 1

参数1 --  
参数2 --  
参数3 \* 127.0.0.1

输出参数

参数名	参数值
执行结果	--

异常信息

这里是异常信息测试，This is exception.

操作按钮：刷新、+、-

接着查看 APP 组件类型日志，确定是都是 ESB API 调用异常。

The screenshot shows a log search interface with the following details:

- 环境 (Environment): 全部
- 类型 (Type): 组件调用 (Component Call), highlighted with a red box.
- 日志级别 (Log Level): 全部
- 函数名 (Function Name): 请选择 (Select)
- 信息 (Information): 请输入查询字符 (Input search character)
- 精确查找 (Exact Match) (checkbox checked)
- 搜索 (Search) button
- 导出前1000条 (Export first 1000 items) button

The search results table has columns: 环境 (Environment), 类型 (Type), 日志级别 (Log Level), 时间 (Time), and 信息 (Information). One result is shown:

环境	类型	日志级别	时间	信息
正式	Compon ent	ERROR	2018-06-10 16:56:02+08:0 0	组件返回错误信息:缺少参数bk_token, request_id=52bf454abcaf4794b8331555 171530de&nb...

Below the table, detailed information is provided:

- 文件 (File): ./blueking/component/base.py
- 函数 (Function): \_\_call\_\_
- 行号 (Line Number): 86
- 日志信息 (Log Message): 组件返回错误信息:缺少参数bk\_token, request\_id=52bf454abcaf4794b8331555171530de url=http://... 30/ap  
i/c/compapi/bk\_login/get\_user/ params={} data={}

## 提交代码

执行 `python manage.py collectstatic -noinput`，然后就可以提交代码并打包发布了。

## 标准插件开发规范

- 分组命名规则是“系统名(系统英文缩写)”，如“作业平台(JOB)”。
- 标准插件编码(code)使用下划线方式，规则是“系统名\_接口名”，如 job\_execute\_task。
- 后台类名使用驼峰式，规则是“标准插件编码+继承类名”，如 JobExecuteTaskService。
- 前端 JS 文件目录保持和系统名缩写一致，JS 文件名保持和标准插件编码一致。
- 参数 `tag_code` 命名规则是“系统名\_参数名”，这样可以保证全局唯一；长度不要超过 20 个字符。

## 需求自助化:测试自助调整验收环境

### 情景

测试同学经常找运维同学调整测试环境的配置，比如调整不同的角色或代币（本文以调整角色为例）来验证功能是否符合预期，一来让运维时间碎片化，二来交付给需求方的时间不固定，需求方也不满意，影响整体效率。

需要一种让需求方自助操作的入口，简化输入，关注输出，解脱运维的同时提升需求方的满意度。

### 前提条件

- 准备一个 `测试` 角色，并在蓝鲸配置平台中将其添加到业务的 `测试人员` 角色中
- 准备调整角色的标准运维流程模板

## 操作步骤

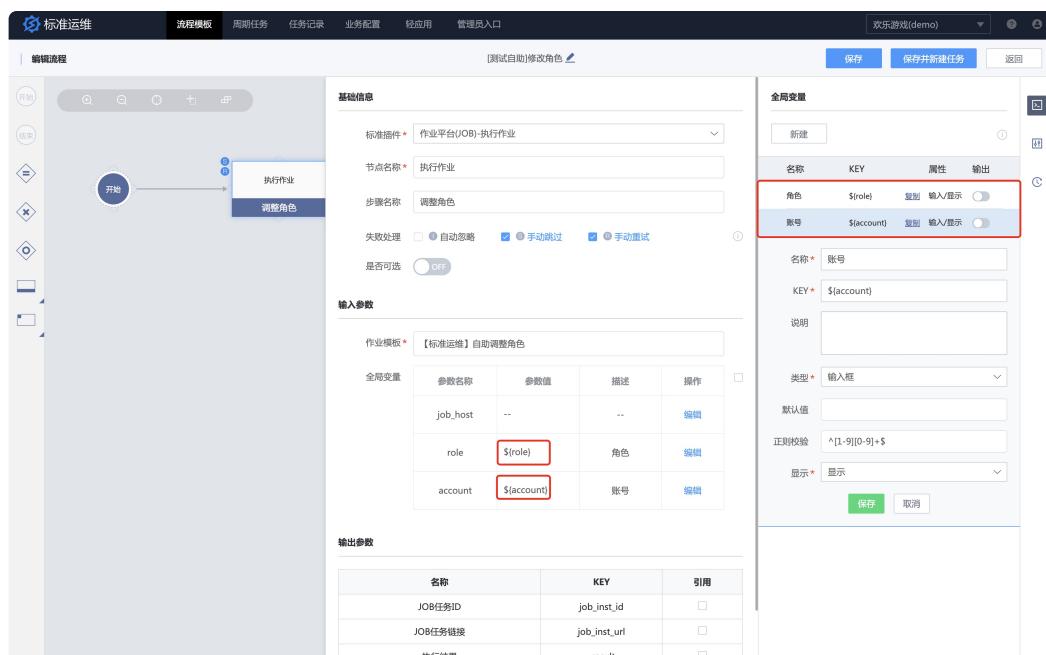
- 新建标准运维流程模板
- 新建自助调角色轻应用
- 测试轻应用

## 新建标准运维流程模板

### 新建流程模板

调整用户角色的逻辑很简单，一般作业平台的一个作业即可满足，此处不展开介绍如何创建作业。

新建流程模板，如下。



由于轻应用重点之一是 **简化输入**，所以尽可能方便需求方填写参数。此处有 2 个参数：**角色** 和 **账号**。

角色一般可以枚举，所以使用 **下拉框**，简化输入。

名称	KEY	属性	输出
角色	\${role}	复制	输入/显示 <input checked="" type="checkbox"/>

右侧

KEY \*

说明

类型 \*

配置  自定义  远程数据源

```
[{"text": "列兵1", "value": 0}, {"text": "列兵2", "value": 1}, {"text": "三等兵", "value": 2}, {"text": "二等兵", "value": 3}, {"text": "一等兵", "value": 4}, {"text": "上等兵1", "value": 5}, {"text": "上等兵2", "value": 6}, {"text": "下士1", "value": 7}, {"text": "中士1", "value": 8}]
```

单选  多选

请输入下拉框的默认值，单选为 value 的格式，多选为 value,value,... 的格式

显示 \*

账号这里是数字，使用 **输入框**，同时加上正则，帮用户纠错。

11 账号 \${account} 复制 输入/显示 ×

名称 \* 账号

KEY \* \${account}

说明

类型 \* 输入框 ▾

默认值

正则校验 ^ [1-9] [0-9] + \$

显示 \* 显示 ▾

保存 取消

保存任务流程，并执行任务，验证任务流程可调整角色。

### 给测试账号授权

选择流程模板，对 测试 账号 ceshi 赋予 新建、执行 任务权限。

The screenshot shows the 'Business Flow Management' interface. In the center, a modal window titled 'Permissions Management' is open. It contains three dropdown menus: 'New Task Permission' (ceshi(ceshi)), 'Accept Task Permission' (请选择), and 'Execute Task Permission' (ceshi(ceshi)). Each of these dropdowns is highlighted with a red box. At the bottom of the modal are two buttons: 'Confirm' (确定) and 'Cancel' (取消).

在 **业务配置** 中设置任务真正的执行者（比如执行作业平台的作业），因为测试没有运维权限。

The screenshot shows the 'Business Configuration' interface. In the center, there is a section titled 'Executor Configuration'. Within this section, there is a field labeled 'Task Executor' containing the value 'blueking', which is also highlighted with a red box. Below this field is a switch labeled '强制生效' (Force Effect) with the value 'OFF'. At the bottom right of the configuration section is a green 'Save' button.

## 新建自助调角色轻应用

点击 **轻应用** 导航栏，按提示新建轻应用。

The screenshot shows the 'Create Light Application' interface. It includes the following fields:

- 流程模板: (selected: 测试自助修改角色)
- 应用名称: 自助调角色
- 执行方案: (请选择)
- 应用简介: 产品或测试可调整测试环境的角色，快速验证业务功能。
- 应用LOGO: A placeholder area with the text 'LVS'.

At the bottom of the form are two buttons: 'Confirm' (确定) and 'Cancel' (取消).

## 测试轻应用

使用测试账号 ceshi 登录蓝鲸，找到刚才创建的 **自助调角色 SaaS**。

The screenshot shows the Blue Whale Application Panel interface. At the top, there is a navigation bar with the logo '蓝鲸智云 | 工作台' on the left and '应用面板' and '开发者中心' on the right. Below the navigation bar is a grid of service icons:

应用面板			
	配置平台		作业平台
	故障自愈		日志检索
	蓝鲸监控		节点管理
	标准运维		自助调角色

The '自助调角色' icon is highlighted with a red border.

点击 **自助调角色 SaaS**

The screenshot shows the '自助调角色' SaaS configuration interface. The top navigation bar includes tabs for '标准运维' (Standard Operations), '新建任务' (New Task), and '任务记录' (Task Record). A user account 'ceshi' is logged in.

The main area is divided into several sections:

- 任务信息**:
  - 任务名称: [测试启动]修改角色\_20190723170401
  - 流程类型: 默认任务流程 (selected)
  - 参数填写
  - 任务执行
- 参数信息**:
  - 角色: 中士1
  - 账号: 33333333

At the bottom, there are '上一步' (Previous Step) and a green '下一步' (Next Step) button.

执行任务



至此，测试同学自助调整测试环境的轻应用已配置完毕，后续测试同学直接打开[自助调角色](#) SaaS 即可完成需求。

此外，如产品和测试需要调整配置，DBA 需要 AWR 报告，测试需要添加白名单等等运维的日常需求，均可以通过轻应用方式解决，简化输入，关注输出，在解脱运维的同时，还能提升需求的满意度。

## 快速构建 Nginx 集群

### 情景

传统的 Nginx 集群要先部署多个 Nginx 节点，然后通过 `upstream` 统一一个入口提供给用户访问。

该过程操作繁琐，接下来看 BCS（容器管理平台）如何通过 容器调度 (以 K8S 编排为例，BCS 同时还支持 Mesos) 快速构建 Nginx 集群。

### 前提条件

- K8S 基本概念，包含 Deployment、Services。
- 完成 BCS 部署
- 准备 2 台云主机：4 核 8 G，不低于 CentOS 7，K8s Master 和 Node 各 1 台
- 完成上述 2 台云主机的 Agent 安装，并分配至 CMDB 业务下

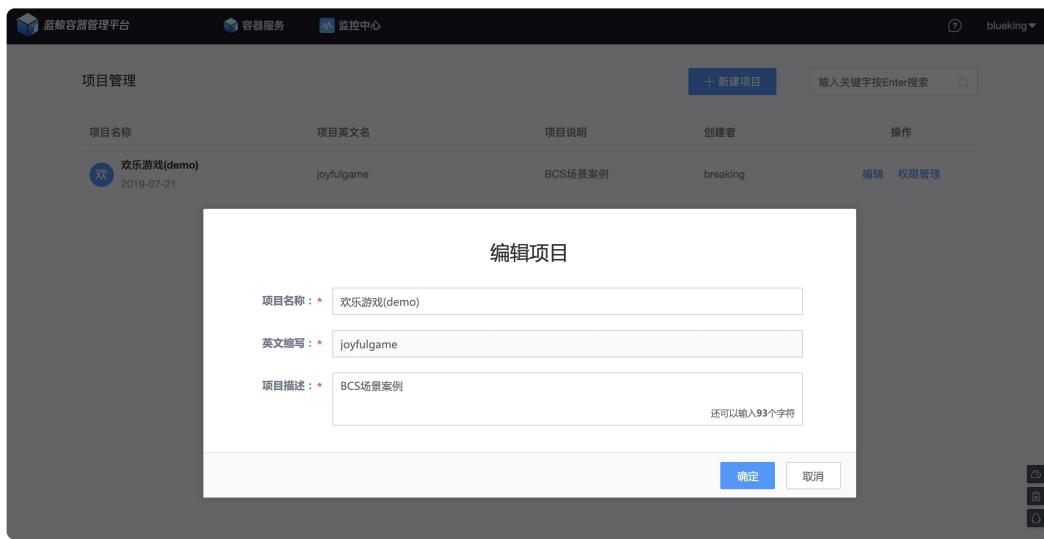
## 操作步骤

- 新建集群
- BCS 快速构建 Nginx 集群

### 新建集群

#### 启用容器服务

在 BCS 首页，点击 **新建项目**，如 **欢乐游戏(demo)**。



然后选择容器编排类型为 **Kubernetes**，关联前提条件中提到的 CMDB 业务，点击 **启用容器服务**。



### 新建集群

启用容器服务后，进入容器服务欢迎页，点击 **创建容器集群**。

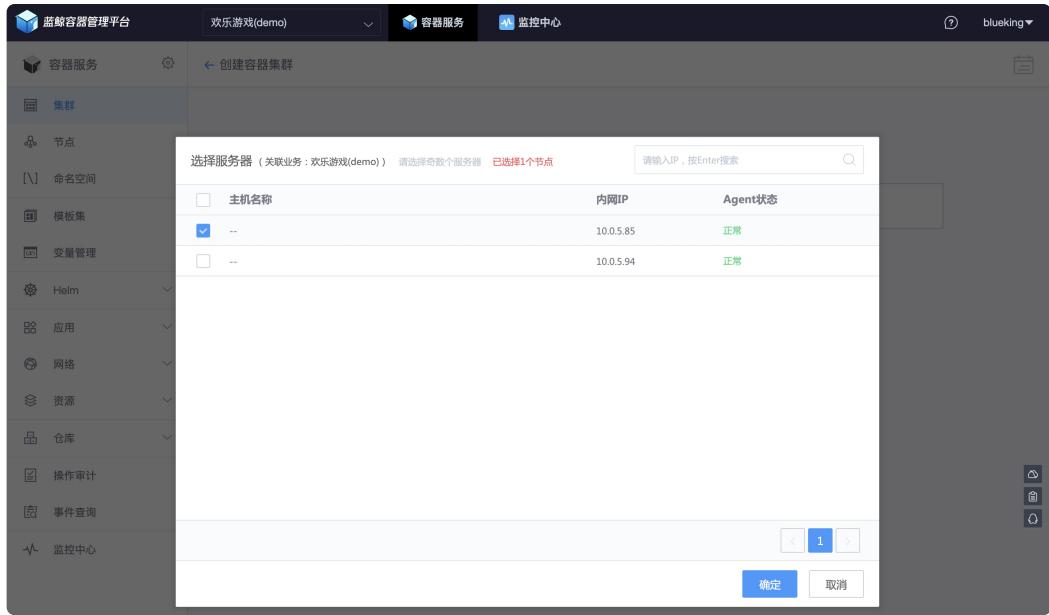
The screenshot shows the BlueKing Container Service Management Platform. The top navigation bar includes '容器服务' (Container Service), '欢乐游戏(demo)', '容器服务' (Container Service), and '监控中心' (Monitoring Center). The left sidebar has a '集群' (Cluster) section selected, containing options like '节点' (Nodes), '命名空间' (Namespaces), '模板集' (Template Sets), '变量管理' (Variable Management), 'Helm', '应用' (Applications), '网络' (Network), '资源' (Resources), '仓库' (Repository), '操作审计' (Audit Operations), '事件查询' (Event Query), and '监控中心' (Monitoring Center). The main content area features a '欢迎使用容器服务' (Welcome to Container Service) banner with a '请点击了解更多' (Click here for more information) link, a '创建容器集群' (Create Container Cluster) button, and a '快速入门指引' (Quick Start Guide) button. To the right, there is a 'BlueKing Container Service Usage Tutorial' section with steps: 1. 新建集群 (Create a new cluster), 2. 集群增加Node节点 (Add Node nodes to the cluster), and 3. 项目镜像管理 (Project Image Management). At the bottom, there are '继续' (Continue) and '关闭' (Close) buttons.

按提示填写集群的基本信息。

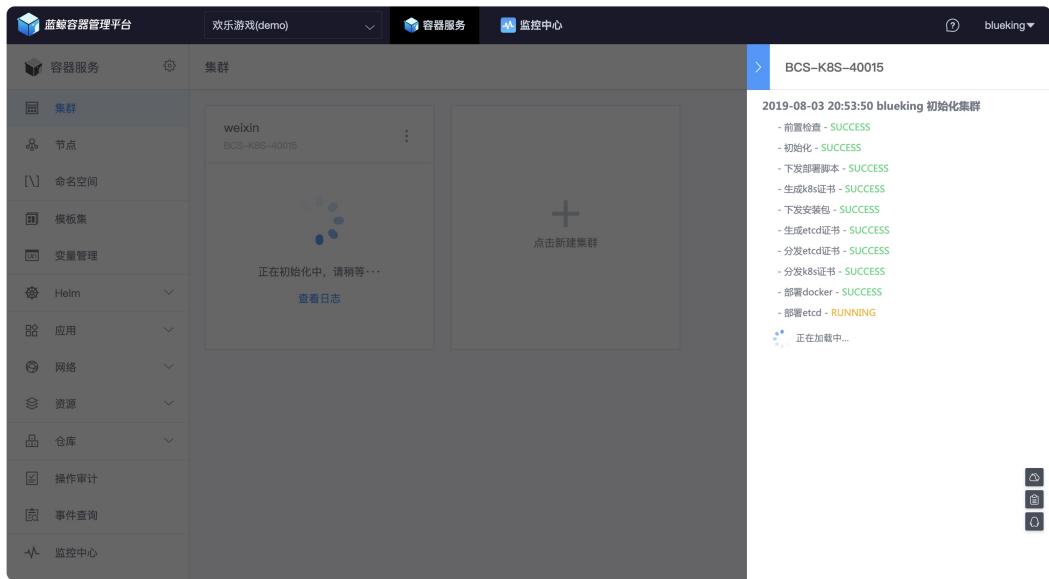
The screenshot shows the 'Create Container Cluster' configuration page. The left sidebar is identical to the previous screen. The main form has fields for '名称' (Name) set to 'weixin', '集群描述' (Cluster Description) set to '微信区', and '选择Master' (Select Master) set to '选择服务器' (Select Server). Below these are tables for '已选服务器' (Selected Servers) and '注意事项' (Notes). The '已选服务器' table shows one server:序号 (Index) 1, IP地址 (IP Address) 10.0.5.85, and 操作 (Operation) with a '移除' (Remove) link. The '注意事项' section contains two checked checkboxes: '服务器将按照系统规则修改主机名' (The server will change its host name according to system rules) and '服务器将安装容器服务相关组件' (The server will install container service related components). At the bottom are '确定' (Confirm) and '取消' (Cancel) buttons.

容器服务的集群划分和 [传统单体应用在 CMDB 中的集群划分](#) 很类似，可以按照 [地域](#)（如华北区）或者 [完全独立的应用集合（微信区）](#) 来划分。

选择 1 台云主机作为 Master。



点击 **确定** 后，集群开始初始化。



点击 **节点管理**

The screenshot shows the BlueKing Container Management Platform interface. The top navigation bar includes tabs for '容器服务' (Container Service), '欢乐游戏(demo)', '容器服务' (Container Service), and '监控中心' (Monitoring Center). The left sidebar has a '集群' (Cluster) section with options like '节点' (Nodes), '命名空间' (Namespaces), '模板集' (Template Sets), etc. The main content area displays a cluster named 'weixin' (BCS-K8S-40015). A context menu is open over the cluster name, with '节点管理' (Node Management) highlighted and surrounded by a red box. Other menu items include '总览' (Overview), '集群信息' (Cluster Information), and '删除' (Delete). Below the cluster name, resource usage statistics are shown: CPU 使用率 (CPU Usage Rate), 内存 使用率 (Memory Usage Rate), and 磁盘 使用率 (Disk Usage Rate), with a 0% value for disk usage.

点击 **添加节点**，按提示节点添加。

This screenshot shows the '节点管理' (Node Management) page for the 'weixin' cluster. The top navigation bar and sidebar are identical to the previous screenshot. The main area shows a table of nodes. A red box highlights the '+ 添加节点' (Add Node) button. The table columns include '主机名/IP' (Host Name/IP), '状态' (Status), '容器数量' (Number of Containers), 'CPU' (CPU), '内存' (Memory), '磁盘IO' (Disk I/O), and '操作' (Operations). One node is listed: '10.0.5.94' which is '初始化中' (Initializing). There is a '查看日志' (View Log) link next to it. At the bottom, there is a pagination control showing '共计1条 每页 5 条' (Total 1 item, 5 items per page).

至此，新建集群完毕。可以看到集群的基础信息。

集群

weixin  
BCS-K8S-40015

CPU使用率 29.13%

内存使用率 14.76%

磁盘使用率 8.53%

另外，在集群的设置(:)下拉菜单中，可以看到集群主要性能指标。

weixin (BCS-K8S-40015)

概述 节点管理 集群信息

CPU使用率 29.13%  
1.17 of 4.00 Shares

内存使用率 14.75%  
1.10GB of 7.46 GB

磁盘使用率 8.53%  
26.43GB of 309.81 GB

节点 使用中 1台 未使用 0台

命名空间 已使用 0个 总量 0个

## BCS 快速构建 Nginx 集群

### 新建命名空间

新建命名空间 **dev**

容器服务 - 命名空间

名称	所属集群	变量	操作
preprod	weixin	--	设置变量值
dev	weixin	--	设置变量值
prod	weixin	--	设置变量值

## 新建模板集

模板集，可以类比为 K8S 中 **Helm** 的 **Charts**，在 K8S 编排中，是 K8S 对象的集合：**Deployment**（无状态）、**StatefulSet**（有状态）、**DaemonSet**（守护进程集）、**Job**（定时任务）、**Configmap**（配置项）、**Secret**（保密字典），具体参见 [模板集使用介绍](#)。

打开菜单 [**模板集**]，新建模板集 **web-nginx**。

容器服务 - 模板集

模板集名称	类型	容器 / 镜像	操作
示例模板集 最新版本: init_version	自定义	container-redis-default / redis:1.0 container-nginx-default / rumpetroll-openresty:0.51 container-rumpetroll-v1 / pyrumpetroll:0.3	案例化 更多

按提示，填写 **Deployment**

容器服务 - web-nginx - 模板集描述

Deployment 1 Service 0 Configmap 0 Secret 0 DaemonSet 0 Job 0 StatefulSet 0 Ingress 0

应用名称: \* web-nginx 实例数量: \* 2 2个副本, Pods 导入YAML

重要级别:  重要  一般  不重要

描述: 请输入256个字符以内

标签: \* app = web-nginx +  添加至选择器

小提示: KBS使用选择器 (spec.selector.matchLabels) 关联资源，必填，且实例化后选择器的值不可变



## 填写 Service

名称: \* web-nginx

关联应用: web-nginx

关联标签: app:web-nginx

Service类型: NodePort

端口映射:

端口名称	端口	协议	目标端口	NodePort
http	8088	TCP	container-	30008

小提示: 同时粘贴多行“键=值”的文本会自动添加多行记录  
Label用于标识Service, 在查询的时候会用到, 比如 kubectl get services -l app=web-nginx -n dev

## 实例化

避免多成员同时编辑, 引起内容或版本冲突, 建议在编辑时, 打开保护功能。 (当前版本号: v1)

## 检查部署效果

在菜单 **网络** -> **Services** 中，找到刚实例化的 Service **web-nginx**

在菜单 **[应用]** -> **[Deployment]** 中可以找到 **web-nginx**

以及其运行指标

The screenshot shows the BlueKing Container Management Platform interface. On the left, there's a sidebar with navigation links: 集群, 节点, 命名空间, 模板集, 变量管理, Helm, 应用, Deployment, DaemonSet, Job, StatefulSet, 网络, 资源, 仓库. The 'Deployment' link is currently selected. In the main content area, there's a summary card for 'web-nginx' with details like 应用名称: web-nginx, 创建时间: 2019-08-08 16:58:19, 更新时间: 2019-08-08 16:58:19, 所在命名空间: dev, 所属集群: weixin. Below this is a chart titled 'CPU使用率' and '内存' showing resource usage over time. Under the 'Pod管理' tab, two pods are listed: 'web-nginx-678bb9c4fb-nxwrf4' and 'web-nginx-678bb9c4fb-m8cf4', both in 'Running' status.

通过访问 `Node+NodePort`，可以查看刚刚部署 Nginx 集群的版本号。

```
[root@ip-10-0-5-94-n-bcs-k8s-40015 ~]# curl 10.0.5.94:30008 -I
HTTP/1.1 200 OK
Server: nginx/1.12.2
Date: Thu, 08 Aug 2019 09:11:42 GMT
```

通过访问 `Service IP + Port`，也可以查看刚部署 Nginx 的版本号。

```
[root@ip-10-0-5-94-n-bcs-k8s-40015 ~]# curl 10.254.11.4:8088 -I
HTTP/1.1 200 OK
Server: nginx/1.12.2
Date: Thu, 08 Aug 2019 09:12:33 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 11 Jul 2017 13:29:18 GMT
Connection: keep-alive
ETag: "5964d2ae-264"
Accept-Ranges: bytes
```

## 应用的滚动升级

### 情景

传统的应用更新方式是停服更新，用户在更新期间 无法使用服务。

接下来，将以 Nginx 从 `1.12.2` 升级 `1.17.0` 为例，看 BCS 中的 滚动更新能力 是如何实现 不停机更新，用户无感知。

### 前提条件

- K8S 基本概念，包含 Deployment、Services。
- 完成 BCS 部署

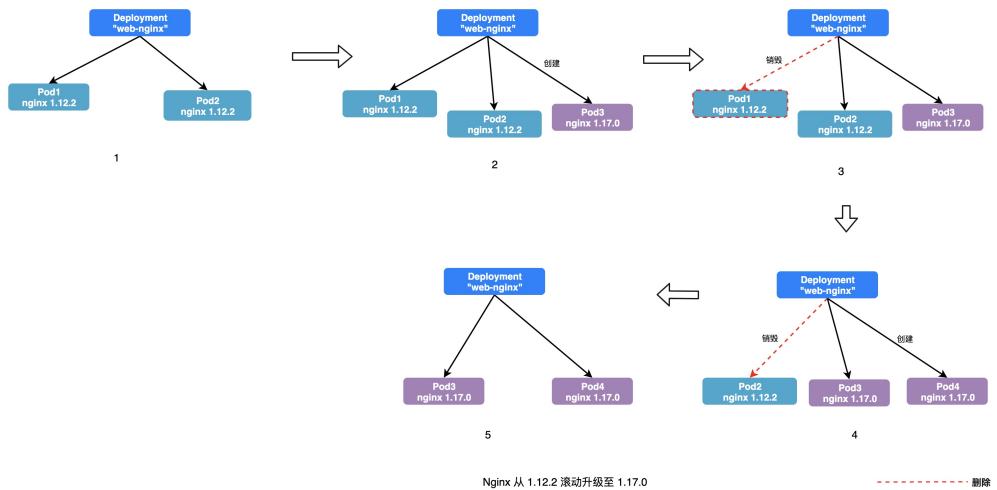
## 操作步骤

- 滚动更新逻辑介绍
- BCS 滚动更新操作指引

### 滚动更新逻辑介绍

滚动更新的逻辑如下图，创建一个新版本的实例（POD），销毁一个旧版本实例（POD），如此滚动，直至线上都是新版本，旧版本已全部销毁。

滚动更新对用户无感知。



### BCS 滚动更新操作指引

#### 推送 NGINX:1.17.0 至镜像仓库

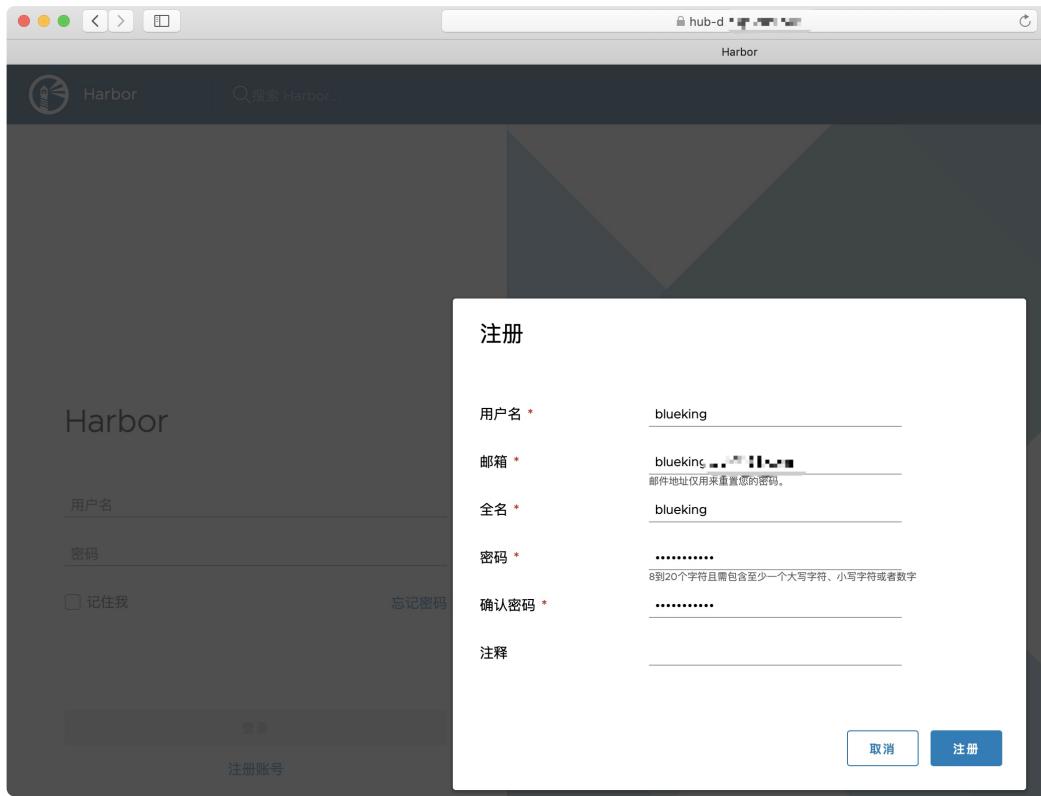
参照 [Harbor 仓库使用指南](#)，将镜像 Nginx:1.17.0 推送至 BCS 公共镜像仓库。

#### 注册镜像仓库账号

在 [部署 BCS](#) 的中控机上获取镜像仓库的访问地址。

```
# source /data/install/utils.fc && echo ${HARBOR_SERVER_FQDN}:${HARBOR_SERVER_HTTPS_PORT}
hub-d.o.*****.com:443
```

登录仓库地址，注册镜像仓库账号。



注册完，登录后可以访问公共仓库。

## 推送 NGINX:1.17.0 至镜像仓库

使用 `docker pull` 从 `hub.docker.com` 拉取镜像 `nginx:1.17.0`。

```
# docker pull nginx:1.17.0
1.17.0: Pulling from library/nginx
fc7181108d40: Pull complete
c4277fc40ec2: Pull complete
780053e98559: Pull complete
Digest: sha256:bdbf36b7f1f77ffe7bd2a32e59235dff6ecf131e3b6b5b96061c652f30685f3a
Status: Downloaded newer image for nginx:1.17.0
```

规范镜像 `tag` 为仓库要求的格式。

```
# docker tag nginx:1.17.0 hub-d.*****.com/public/nginx:1.17.0
```

REPOSITORY	IMAGE ID	CREATED	SIZE	TAG
nginx	719cd2e3ed04	8 weeks ago	109MB	1.17.0
hub-d.*****.com/public/nginx	719cd2e3ed04	8 weeks ago	109MB	1.17.0

推动镜像至 BCS 镜像仓库。

```
# docker push hub-d.*****.com/public/nginx:1.17.0
The push refers to repository [hub-d.*****.com/public/nginx]
d7acf794921f: Pushed
d9569ca04881: Pushed
cf5b3c6798f7: Pushed
1.17.0: digest: sha256:079aa93463d2566b7a81cbdf856afc6d4d2a6f9100ca3bcbeef24ade92c9a7fe
size: 948
```

在镜像仓库中，可以找到刚推送的 **Nginx:1.17.0** 镜像。

Harbor UI screenshot showing the public/nginx repository details. The table lists one image tag:

标签	大小	Pull命令	作者	创建时间	Docker 版本
1.17.0	42.70MB	public/nginx:1.17.0	NGINX Docker Maintainers <docker-ma...	2019/6/11 上午8:03	18.06.1-ce

在 BCS 的 **[仓库菜单]** 中也可以找到。

BCS Container Management Platform screenshot showing the public/nginx repository details. The table lists one image tag:

名称	大小	最近更新时间	地址
1.17.0	42.7 MB	2019-08-08 17:42:25	hub-d.*****.com:443/public/nginx:1.17.0

## 滚动升级 NGINX：从 1.12.2 到 1.17.0

确认当前版本号为 **nginx/1.12.2**

```
[root@ip-10-0-5-94-n-bcs-k8s-40015 ~]# curl 10.0.5.94:30008 -I
HTTP/1.1 200 OK
Server: nginx/1.12.2
Date: Thu, 08 Aug 2019 09:11:42 GMT
```

在【模板集】的【Deployment】页面中，修改【镜像及版本】，将版本从 **1.12.2** 修改为在推送 Nginx:1.17.0 至镜像仓库中上传的 Nginx 新镜像 **1.17.0**。

The screenshot shows the BlueKing Management Platform interface. On the left, there's a sidebar with various management sections like Helm, Application, Network, Resource, and Catalog. The main area is titled 'Deployment' and shows a single entry for 'web-nginx'. Under 'Deployment', there are fields for '应用名称' (app name) set to 'web-nginx', '实例数量' (instance count) set to '2', and '重要级别' (importance level) set to '一般' (General). Below this, there's a '标签' (Labels) section where 'app' is mapped to 'web-nginx'. In the '更多设置' (More Settings) tab, under 'Image and Version', the '镜像及版本' (Image and Version) field is set to 'public/nginx 1.17.0'. A red box highlights this field.

点击【更多设置】，了解默认滚动升级的更新策略。

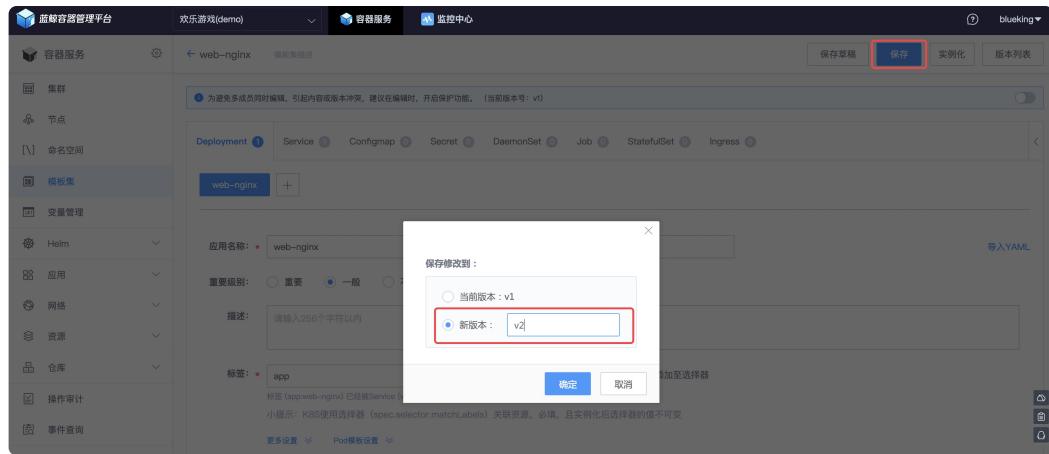
This screenshot shows the 'More Settings' tab for the deployment configuration. Under the 'Update Strategy' section, there are three key parameters: 'maxUnavailable' (set to 1), 'maxSurge' (set to 0), and 'minReadySeconds' (set to 0). These parameters define how many pods can be unavailable during a rolling update and how many new pods can be created simultaneously. A red box highlights the 'maxUnavailable' field.

“

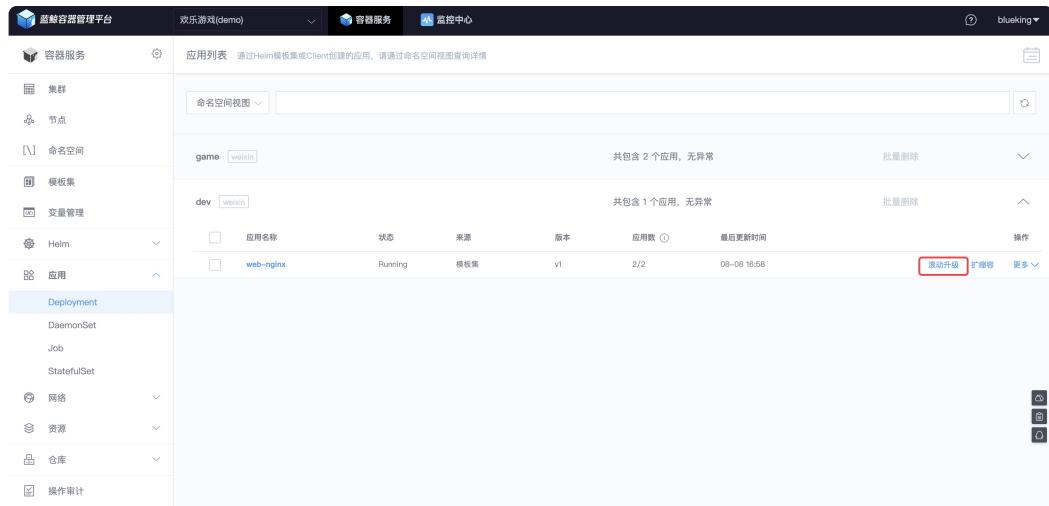
- **maxUnavailable**：滚动升级期间，考虑应用容量，不可用 Pod 的数量上限
- **maxSurge**：滚动升级期间，考虑集群资源，超出期望 Pod 的数量上限
- **minReadySeconds**：滚动升级期间，考虑可用性，探测 Pod 正常后转为可用的时间

”

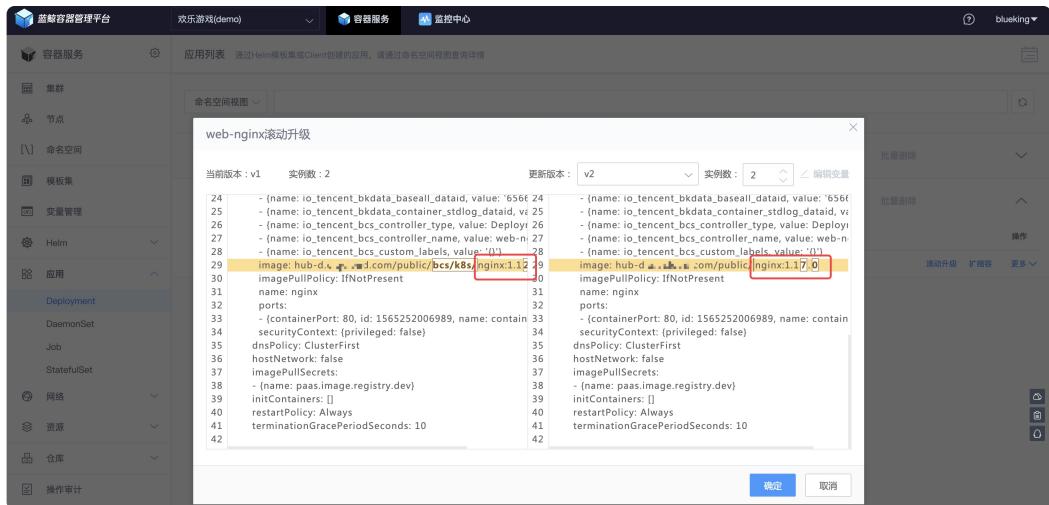
修改完镜像的版本后，接下来【保存】模板集，填写【新版本】的版本号。



接着，开始滚动升级。点击菜单【应用】->【Deployment】，找到 web-nginx 应用，点击【滚动升级】。



可以看到，差异点是 `images` 从 `nginx:1.12.2` 调整为 `nginx:1.17.0`



点击【确定】滚动升级后，正在更新。

通过右下角的【Web console】可以通过命令行的方式获取实例(POD)的基础信息。

以下是更新前后的对比

```
root:~$ kubectl get pods -n dev -o wideNAME                                READY   STATUS    R  
ESTARTS   AGE     IP          NODE  
web-nginx-678bb9c4fb-m8cf4   1/1     Running   0          134m   172.32.1.18   ip-10-0-  
5-94-n-bcs-k8s-40015   <none>  
web-nginx-678bb9c4fb-nxwf4   1/1     Running   0          134m   172.32.1.17   ip-10-0-  
5-94-n-bcs-k8s-40015   <none>  
  
root:~$ kubectl get pods -n dev -o wide  
NAME          READY   STATUS    RESTARTS   AGE     IP          NODE  
NOMINATED NODE  
web-nginx-f95ffc78d-cxhrq   1/1     Running   0          21s    172.32.1.20   ip-10-0-5-  
94-n-bcs-k8s-40015   <none>  
web-nginx-f95ffc78d-pqtcj   1/1     Running   0          14s    172.32.1.21   ip-10-0-5-  
94-n-bcs-k8s-40015   <none>
```

可以看到 Nginx 版本已经从 1.12.2 更新为 1.17.0

```
[root@ip-10-0-5-94-n-bcs-k8s-40015 ~]# curl 10.0.5.94:30008 -I  
HTTP/1.1 200 OK  
Server: nginx/1.17.0
```

## 应用的蓝绿发布（原：研发测试环境管理）

### 情景

传统的应用更新方式是停服更新，用户在更新期间无法使用服务。

接下来，将以 Nginx 从 1.12.2 升级 1.17.0 + 程序代码（index.html 的内容从 Nginx 默认页更新为 1.17.0）为例，看 BCS 中的蓝绿发布能力是如何实现不停机更新，用户无感知。

### 前提条件

- K8S 基本概念，包含 Deployment、Services；本节教程新增概念：ConfigMap、Ingress、Ingress Controllers。
- 完成 BCS 部署

### 操作步骤

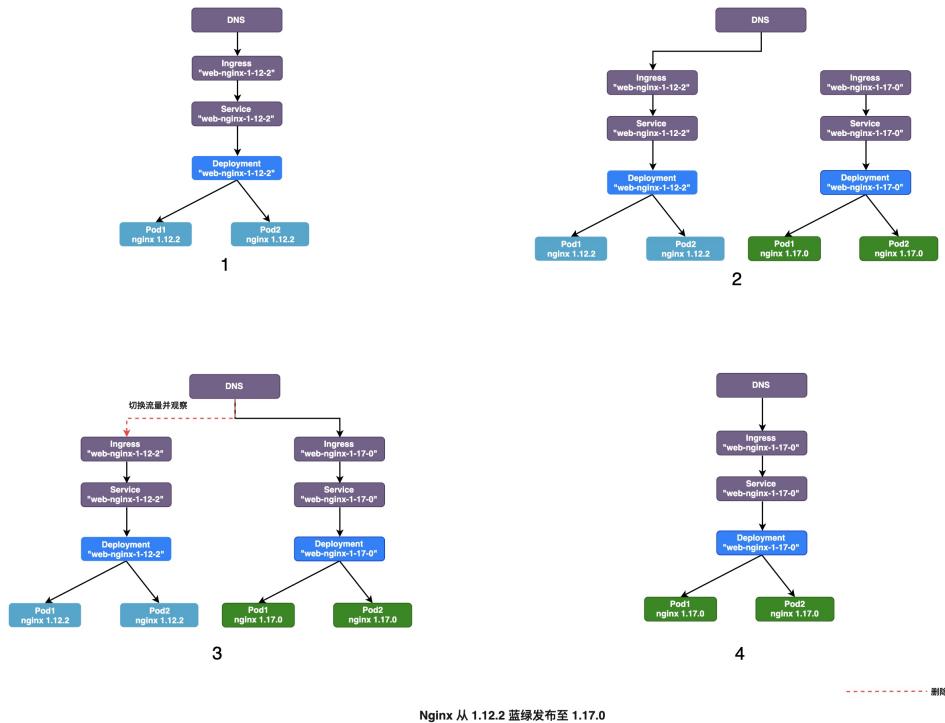
- 应用的蓝绿发布逻辑介绍
- 使用 K8S 资源准备版本
- 使用 K8S 资源准备新版本

- 切换流量并观察

## 蓝绿发布逻辑介绍

### 发布逻辑示意图

蓝绿发布，即准备当前运行版本 和 新版本 两组实例，正式发布的时候，修改服务的域名的 DNS 记录将，将其指向新版本的 Ingress 指向的地址。



### 版本更新流程中引入的对象

以 Nginx 从 `1.12.2` 升级 `1.17.0` + 程序代码 (`index.html` 的内容从 Nginx 默认页更新为 `1.17.0`) 为例，使用以下几个新的对象：

- 程序代码或可执行文件 (`index.html`) : Docker 镜像
- 程序或运行环境配置 (`nginx.conf`) : ConfigMap
- 负载均衡器 (LoadBalancer) + Ingress : 用户接入和负载均衡

其中 Deployment、Service 不再赘述。

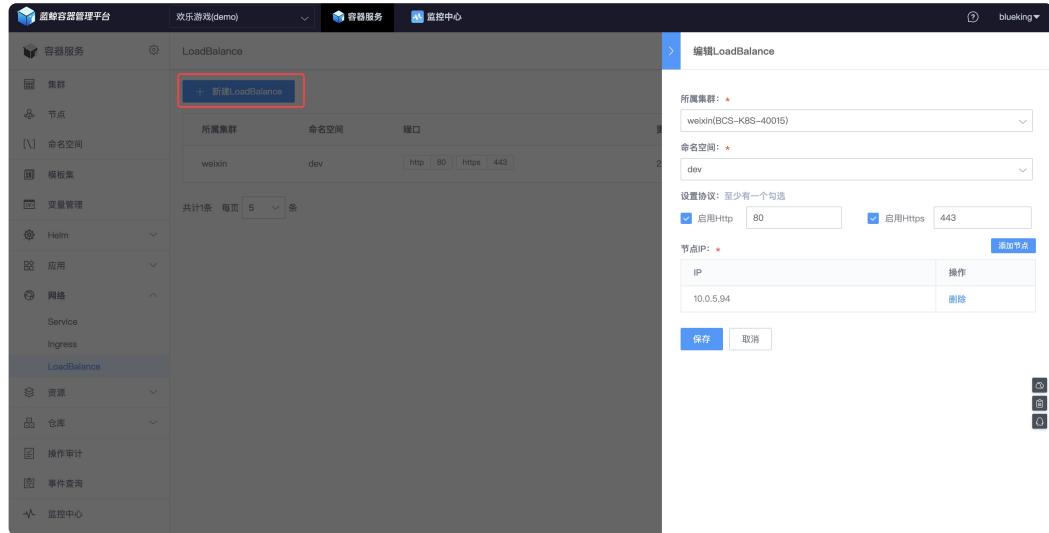
## 使用 K8S 资源准备版本

### 新增 LOADBALANCER

Ingress 是 K8S 中描述用户接入的对象之一，需要配合 LB 应用才能对外提供访问。

在 BCS 中，LoadBalancer 背后的技术是 K8S 维护的 **Nginx Ingress Controller**。

选择菜单【LoadBalancer】，点击【新建 LoadBalancer】，选择一个节点作为 LB 对外提供网络接入服务。



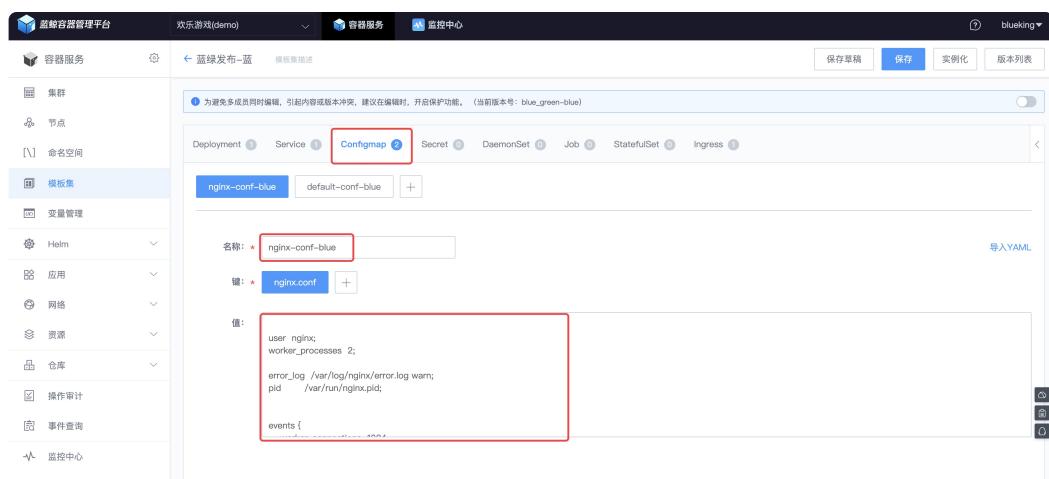
“

建议业务 Pod 不调度至 LB 所在的节点，可使用 节点亲和性（nodeAffinity）实现。

”

## CONFIGMAP：存放 NGINX.CONF

在【模板集】菜单中，选择【Configmap】，新建 nginx.conf 和 default.conf 两个 configmap，实现应用程序和配置的解耦。



在【Deployment】中，填写 名称、标签（Label）、容器镜像、挂载卷（挂载 Configmap）。

The screenshot shows the BlueKing Container Management Platform interface. On the left, there's a sidebar with various navigation options like Cluster, Node, Namespace, Template Set, Helm, Application, Network, Resource, Library, Audit, and Event Query. The main area is titled "Blue Green Release - Blue".

**Deployment Configuration:**

- Application Name: web-nginx-blue
- Instance Count: 2
- Importance Level: General (selected)
- Description: Please enter 255 characters or less.
- Labels: app = web-nginx-blue
- Annotations: K8S use selector (spec.selector.matchLabels) to associate resources, mandatory, and after simplification, the value of the selector cannot be empty.
- Volume Mounts (ConfigMap):
  - Container: nginx-conf-blue (selected)
  - Volume: nginx-conf-blue
  - Path: /etc/nginx/nginx.conf
  - Container: default-conf-blue (selected)
  - Volume: default-conf-blue
  - Path: /etc/nginx/conf.d/default.conf

**Service Configuration:**

- Container: nginx
- Type: Container (selected)
- Description: Please enter 255 characters or less.
- Image and Version: public/bcs/k8s/nginx 1.12.2-autoreload
- Port: container-port 80
- Mounts (挂载卷):
  - Container: nginx-conf-blue (selected)
  - Volume: nginx.conf
  - Path: /etc/nginx/nginx.conf
  - Container: default-conf-blue (selected)
  - Volume: default.conf
  - Path: /etc/nginx/conf.d/default.conf

Annotations: Only mount one file, prevent mounting entire directory.

#### ● 创建 Service

在【Service】中关联 Deployment 以及服务名称、暴露的端口。

The screenshot shows the BlueKing Container Management Platform interface. On the left, there's a sidebar with navigation items like '容器服务', '集群', '节点', '命名空间', '模板集' (which is selected), '变量管理', 'Helm', '应用', '网络', '资源', '仓库', '操作审计', '事件查询', and '监控中心'. The main content area has tabs at the top: 'Deployment', 'Service' (selected), 'ConfigMap', 'Secret', 'DaemonSet', 'Job', 'StatefulSet', and 'Ingress'. The 'Ingress' tab is highlighted. Below the tabs, there's a form for creating an Ingress. The '名称' (Name) field is set to 'web-nginx-blue'. The '关联应用' (Associated Application) dropdown is set to 'web-nginx-blue'. Under 'Service类型' (Service Type), 'ClusterIP' is selected. The 'ClusterIP' field contains '请输入 ClusterIP' and '不填或None'. The '端口映射' (Port Mapping) section shows a mapping from port 80 (HTTP) to port 80 (TCP) of the service. The '标签管理' (Label Management) section shows a label 'app' mapped to the service. A note at the bottom says '小提示：同时粘贴多行“键=值”的文本会自动添加多行记录'. At the top right, there are buttons for '保存草稿', '保存' (Save), '实例化' (Instantiate), and '版本列表'.

### ● 新建 Ingress

在【Ingress】中填写【主机名】、【路径】以及绑定【Service】。

This screenshot shows the same interface as the previous one, but the 'Ingress' tab is now active. The '名称' field still contains 'web-nginx-blue'. Below it, there are 'TLS设置' and '更多设置' buttons. The main configuration area is titled '在 K8S Nginx Ingress Controller 中，背后对应 nginx.conf 的 server\_name' and '基础信息'. It shows the '主机名' (Host) as 'blue.bk.tencent.com', 'Service 名' (Service Name) as 'web-nginx-blue', and 'Service 端口' (Service Port) as '80'. The '路径组' (Path Group) section shows a path '/> web-nginx-blue:80'. The right side of the screen has the same set of buttons as the first screenshot.

## 实例化模板集

保存模板集后，实例化模板集。

可以看到对应资源已生成好。

### • Deployment

### • Service

### • Ingress

名称	所属集群	命名空间	来源	创建时间	更新时间	更新人	操作
web-nginx-green	weixin	dev	模板集	2019-08-16 16:40:11	2019-08-16 16:40:32	blueking	<a href="#">查看</a> <a href="#">删除</a>
web-nginx-blue	weixin	dev	模板集	2019-08-16 16:20:51	2019-08-16 16:21:32	blueking	<a href="#">查看</a> <a href="#">删除</a>

修改域名解析或修改 PC 上 hosts 文件（Mac 下路径为 /etc/hosts），将 Ingress 中配置的主机名解析到 LoadBalancer 中节点的外网 IP，然后打开浏览器访问。

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 [nginx.org](http://nginx.org).  
Commercial support is available at [nginx.com](http://nginx.com).

Thank you for using nginx.

以上作为线上环境运行的版本，接下来部署新版本。

## 使用 K8S 资源准备新版本

本次新版本参照微服务更新的最佳实践：将应用程序打入 Docker Image，更新 Deployment 中的镜像即更新版本。

## 制作新版本的 DOCKER IMAGE

以 index.html 为应用程序，将其打入镜像，由于新版本的运行时是 Nginx 1.17.0，故以 Nginx 1.17.0 为基础镜像。

- 准备 dockerfile

```
$ ll
total 16
-rw-r--r-- 1 breaking staff 49B 9 10 10:55 dockerfile
-rw-r--r-- 1 breaking staff 40B 9 10 10:54 index.html

$ cat index.html

Welcome to BCS.
```

```
Nginx Version: 1.17.0

$ cat dockerfile
FROM nginx:1.17.0
COPY index.html /usr/share/nginx/html
```

- 构建 Docker Image

```
$ docker build -t bcs_nginx:1.17.0 .
Sending build context to Docker daemon 3.072kB
Step 1/2 : FROM nginx:1.17.0
--> 719cd2e3ed04
Step 2/2 : COPY index.html /usr/share/nginx/html
--> 9e1342027c80
Successfully built 9e1342027c80
Successfully tagged bcs_nginx:1.17.0

$ docker images
REPOSITORY           TAG      IMAGE ID
CREATED             SIZE
bcs_nginx           1.17.0   9e1342027
c80     18 seconds ago    109MB
```

- 推送镜像到仓库

```
$ docker tag bcs_nginx:1.17.0 <Registry_URL>/joyfulgame/bcs_nginx:1.17.0

$ docker push <Registry_URL>/joyfulgame/bcs_nginx:1.17.0
The push refers to repository [<Registry_URL>/joyfulgame/bcs_nginx]
8b2c2f2923c8: Pushed
d7acf794921f: Mounted from joyfulgame/nginx
d9569ca04881: Mounted from joyfulgame/nginx
cf5b3c6798f7: Mounted from joyfulgame/nginx
1.17.0: digest: sha256:b608ac54ca92dd092529f9b86403d3a539eab030103e2e0c1a984787ece448c9
size: 1155
```

“

更多 Docker Image 的构建方法可以参考 [docker-nginx](#)。

”

## 克隆模板集为新版本

由于新版本主要在 Deployment 的镜像版本、Ingress 绑定的主机名，以及这几个 K8S 对象的命名差别（同一个命名空间不允许重名），所以克隆模板集，然后修改即可。

点击【复制模板集】，会克隆一个模板集，命名为：蓝绿发布-绿

- 修改【Deployment】中的名称、标签以及镜像版本。

- 修改【Service】中的名称、关联标签。

- 修改 Ingress，主机名不能上一个版本一样。

最后保存模板集，然后开始实例化模板集。

修改域名解析或修改 PC 上 hosts 文件（Mac 下路径为 /etc/hosts）后，访问 Ingress 中配置的主机名。

Welcome to BCS.

Nginx Version: 1.17.0

## 切换流量并观察

如果是客户端业务，将请求的后端地址指向为新版本的主机名即可，如果客户端不方便更新配置，可以使用 CNAME 将域名指向到新的版本的主机名。

观察一段时间，如果没有异常，删除原版本的实例即可。

The screenshot shows the BCS console interface under the 'Container Service' tab. On the left, there's a sidebar with options like 'Cluster', 'Nodes', 'Namespace', 'Template Set' (which is selected), 'Volume Management', 'Helm', 'Application', 'Network', 'Resource', 'Warehouse', 'Audit Operation', 'Event Inquiry', and 'Monitoring Center'. The main area displays a table for template sets:

模板集名称	类型	操作
蓝绿发布-绿	自定义 最新版本: green	[实例化] [更多]
蓝绿发布-蓝	自定义 最新版本: blue	[实例化] [更多]
web-nginx	自定义 最新版本: nodeport	[实例化] [更多]
示例模板集	自定义 最新版本: init_version	[实例化] [更多]

In the bottom right corner of the 'web-nginx' row, there is a red rectangular box highlighting the 'Delete Instance' button. The status bar at the bottom of the table says '全部数据加载完成'.

- 蓝绿发布的好处：新版本如果发现异常，可以快速的切换到老版本。
- 蓝绿发布的坏处：预备双倍的资源，直到下线老版本。不过如果有云平台，成本可控。

企业可以结合自身实际情况，选择合适的版本发布方式。

## 集成内部语音网关实现电话告警

“ 特别感谢社区用户 **Kevin** 提供该文档.”

”

## 情景

故障处理是运维的几大职能之一，重要告警不能漏告，通过微信、邮件、短信等通知方式无法触达，所以需要使用电话告警。

蓝鲸监控默认已支持腾讯云的语音网关，实现电话告警通知，但如果公司已有第三方(非腾讯云)的电话告警服务，该如何接入呢？

## 前提条件

- 开通好企业内部语音网关
- 掌握 [蓝鲸 SaaS 开发](#)，打开 [腾讯运维开发实战课](#) 马上学习
- 掌握 [蓝鲸 API 网关开发](#)

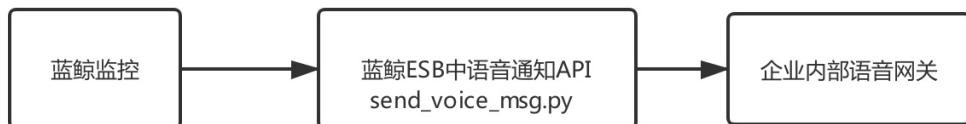
## 操作步骤

- 梳理逻辑
- 代码解读
- 电话告警测试

### 梳理逻辑

对企业内部语音网关封装一个接口，改造蓝鲸 ESB 中语音通知 API 即可。

以下为语音通知的调用逻辑图。



语音通知的调用逻辑图

### 代码解读

#### 封装企业内部语音网关接口

实现内部电话告警接口，并放到 send\_voice\_msg.py 同一层目录下。

本例中的接口：`ums_alarm.ums_tools.send_phone(user_phone_list, content)`

参数为电话电码列表、告警内容，返回值为调用结果( `True` / `False` )和接口的返回消息

## 改造蓝鲸 ESB 中语音通知 API

在蓝鲸 PaaS 所在机器的消息通知代码目录下，修改 `send_voice_msg.py`

```
source /data/install/utils.fc
ssh $APPO_IP
cd /data/bkce/open_paas/esb/components/generic/templates/cmsi/
```

“

注：企业版请将 `bkce` 改成 `bkee`

”

### 获取请求接口的数据

原有 `send_voice_msg.py` 文件已经实现了大部分内容，关键在于 Form 类中的 `handle(self)` 函数，这个函数已实现从请求接口的数据中获取 告警接收人列表`data['user_list_information']`，另外只需要使用 `data['auto_read_message']` 获取告警信息即可。

```
# 公共语音接口/api/c/compapi/cmsi/send_voice_msg/请求参数示例
{
    "bk_app_code": "esb_test",
    "bk_app_secret": "xxx",
    "bk_token": "xxx",
    "auto_read_message": "This is a test",
    "user_list_information": [
        {
            "username": "admin",
            "mobile_phone": "1234567890",
        }
    ]
}
```

### 调用企业内部语音告警接口

修改 `#TODO: can be updated` 之后的部分，全部注释掉。

使用上一步得到的告警信息 `data['auto_read_message']` 和上面接口中得到 `data['user_list_information']`，根据实际情况组装后，以参数形式调用企业内部语音告警接口即可。

```
# TODO: can be updated
# -----实现下面的代码-----
# 取告警内容
v_content = data['auto_read_message']
# 获取用户手机列表
user_phone_list = [user_list['mobile_phone'] for user_list in data['user_list_information']]
# 调用接口，发送告警
ret, msg = ums_tools.send_phone(user_phone_list, v_content.encode('utf-8'))
if ret == True:
    result = {
        'result': True,
```

```
        'data': msg,
        'message': 'OK'
    }
else:
    result = {
        'result': False,
        'data': msg,
        'message': 'Send voice error'
    }
self.response.payload = result
```

## 重启 PaaS

蓝鲸中控机上重启 PaaS，使代码生效

```
/data/install/bkcec stop paas
/data/install/bkcec start paas
```

## 电话告警测试

配置电话告警策略，触发告警，验证结果。

> 主机监控

\* 检测规则  提醒  预警  致命

预警

添加算法

静态阈值

当前值   % 时, 触发告警

高级选项

通知方式  分级告警

告警方式  发生告警  告警恢复

\* 通知方式        admin(admin)

\* 通知角色  主负责人  备份负责人  其他

通知时间段  00:00 至 23:59

告警产生后, 手机将收到语音告警电话。

{% video %}media/send\_voice\_msg.mp3{% endvideo %}

不漏掉任何一个重要的告警。

## 概述

敬请期待

## 概述

敬请期待

# 故障自愈

故障自愈是行业领先的"故障自动化处理"解决方案，提升企业的服务可用性和降低故障处理的人力投入，实现故障自愈从"人工处理"到"无人值守"的变革！

{% video %}media/fta\_brand\_video.mp4{% endvideo %}

通过自动化处理节省人力投入，通过预定的恢复流程让恢复过程更可靠，通过并行分析达到更快的故障定位和恢复。

一句话概括：实时发现告警，预诊断分析，自动恢复故障，并打通周边系统实现整个流程的闭环。

## 蓝鲸监控告警自动处理

### 情景

故障处理是运维的职能之一，人工登录服务器处理告警，存在 2 个问题：**故障处理效率低** 和 **操作疏忽时可能影响生产环境**，例如删除文件输入绝对路径时，在根目录和日志目录间误敲空格，导致根目录删除。

接下来通过“蓝鲸监控的进程告警接入故障自愈”这个案例，来了解故障自愈是如何解决这 2 个痛点。

### 前提条件

- 蓝鲸配置平台纳管了主机
- 蓝鲸配置平台纳管了进程
- 作业平台新建一个作业

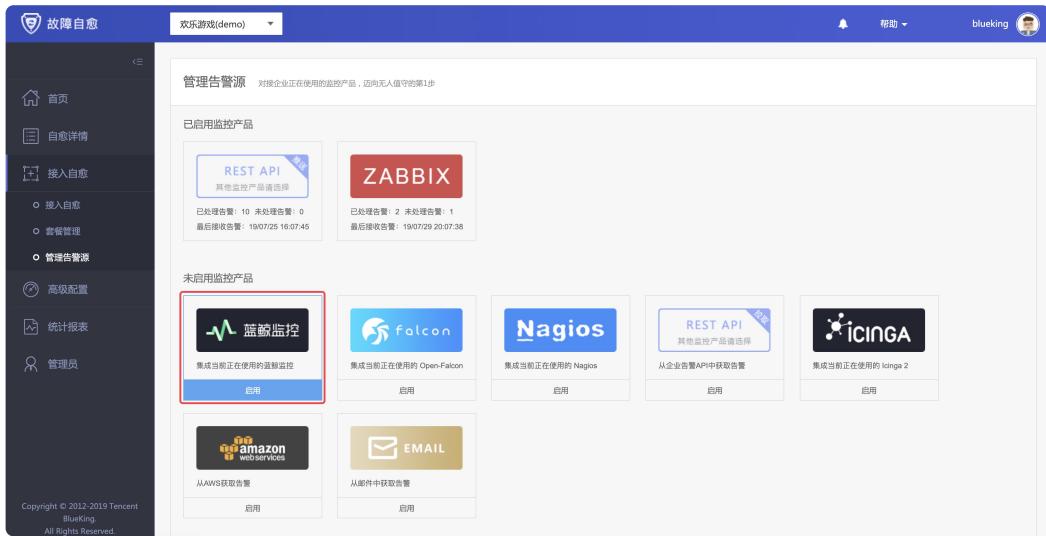
术语解释 - **自愈套餐**：告警的处理动作，如拉起进程的作业； - **自愈方案**：关联 告警 和 处理动作的一个组合；

### 操作步骤

- 启用蓝鲸监控告警源
- 接入自愈方案
- 自愈测试

#### 启用蓝鲸监控告警源

在菜单 [接入自愈] -> [管理告警源] 中，启用 蓝鲸监控。



The screenshot shows the 'Management of Alert Sources' page in the BlueKing console. Under '已启用监控产品' (Enabled Monitoring Products), 'ZABBIX' and 'REST API' are listed. Under '未启用监控产品' (Not Enabled Monitoring Products), '蓝鲸监控' (BlueKing Monitoring) is highlighted with a red box and has a blue '启用' (Enable) button. Other options like 'falcon', 'Nagios', 'Amazon Web Services', and 'EMAIL' are also shown.

## 接入自愈方案

在菜单 [接入自愈] 中，点击 **接入自愈**，告警类型选择 **[主机监控]进程端口**，模块选择 **存储层**，因为不同类型服务器拉起进程的作业不一样。

点击新建 **自愈套餐** 的按钮，准备新建拉起进程的作业。



The screenshot shows the 'Create Recovery Plan' configuration page. Step 1: '告警类型' is set to '[主机监控] 进程端口'. Step 2: '模块' is set to '存储层'. Step 3: '自愈套餐' is set to '请选择处理告警的自愈套餐'. Under '通知方式' (Notification Methods), '开始时' includes WeChat, Email, SMS, and Phone. '成功时' includes WeChat, Email, and Phone. '失败时' includes WeChat, Email, and Phone. Notifiers are listed as '业务运维: blueking/admin'.

在套餐中，套餐类型选择 **作业平台**，新建 **启动MariaDB的作业**。

点击 **新建作业的按钮** 后，跳转至作业平台，在菜单 **[作业执行] -> [新建作业]** 中，新建如下作业：

```

# Check
ps -ef | grep -i mysqld
netstat -ntlp | grep -i 3306

# Start MariaDB
systemctl start mariadb || job_fail "start mariadb fail"

# Check
ps -ef | grep -i mysqld
netstat -ntlp | grep -i 3306
netstat -ntlp | grep -i 3306 || job_fail "mariadb not listen 3306"

job_success "start mariadb succ"

```

保存 **启动MariaDB** 的套餐后，自动回到接入自愈的页面，保存自愈方案即可。

The screenshot shows the '故障自愈' (Fault Recovery) module in the BlueKing platform. In the '自愈场景' (Recovery Scenario) section, a new scenario is being configured for a '进程端口' (Process Port) alert. The '自愈套餐' (Recovery Plan) is set to '启动MariaDB' (Start MariaDB). The '通知方式' (Notification Method) includes WeChat, Email, and SMS for start, success, and failure events. The '通知人' (Notifier) is set to '业务运维: blueking:admin'. A note at the bottom states: '注: 需要提前在蓝鲸控制台的【用户管理】填写联系方式' (Note: You need to pre-fill contact information in the User Management section of the BlueKing Control Panel).

回到接入自愈列表，在列表中可以找到刚刚创建的自愈方案。

The screenshot shows the '接入自愈' (Access Recovery) list. It displays five recovery plans corresponding to different alert types: REST默认分类 (REST Default Category), 磁盘容量(\%fs.) (Disk Capacity (%fs.)), 进程端口 (Process Port), 5分钟平均负载 (5-minute Average Load), and CPU单核使用率 (CPU Single-core Usage Rate). The '启动MariaDB' plan is highlighted with a red box.

告警类型	告警数量	平台	集群	模块	自愈套餐	告警源	自愈方案	方案来源	启用	操作
REST默认分类	10	(所有)	(所有)	(所有)	默认测试套餐	REST API监控	REST API测试	人工配置	是	
磁盘容量(\%fs.)	2	(所有)	(所有)	接入层	接入层日志清理	Zabbix监控	接入层日志清理	人工配置	是	
进程端口	0	(所有)	(所有)	存储层	启动MariaDB	蓝鲸监控	启动MariaDB	人工配置	是	
5分钟平均负载	0	(所有)	(所有)	(所有)	[待挂] 发送C...	蓝鲸监控	5分钟平均负载...	系统推荐	否	
CPU单核使用率	0	(所有)	(所有)	(所有)	[待挂] 发送C...	蓝鲸监控	CPU单核使用率...	系统推荐	否	

## 自愈测试

接下来将停止 MariaDB 进程，来验证是否可以自动启动进程，以恢复 DB 服务。

```
# ps -ef | grep -i mysqld
mysql      926      1  0 10:47 ?        00:00:00 /bin/sh /usr/bin/mysqld_safe --basedir=
/usr
mysql     1159    926  0 10:47 ?        00:00:09 /usr/libexec/mysqld --basedir=/usr --da
tadir=/var/lib/mysql --plugin-dir=/usr/lib64/mysql/plugin --log-error=/var/log/mariadb/
mariadb.log --pid-file=/var/run/mariadb/mariadb.pid --socket=/var/lib/mysql/mysql.sock
root     16763   7429  0 19:40 pts/1    00:00:00 grep --color=auto -i mysqld
# systemctl stop mariadb
```

稍等片刻，收到蓝鲸监控的进程端口告警。

在故障自愈的自愈详情中，找到了该条告警的自愈记录，耗时 21 秒。

跳转到作业平台的执行历史，可以看到 MariaDB 已经启动成功。

回到监控的事件中心，可以看到 告警已经恢复。

告警自动处理，如此简单。

以上为主机监控的告警自动化处理，其他类型告警请参考对应文档：[组件监控告警自动化处理](#)、[自定义采集的告警自动化处理](#)。

故障自动处理是把双刃剑，需要考虑因为网络波动等场景导致的假告警，这时可以用到故障自愈的[异常防御需审批](#)功能。具体请参照[故障自愈的收敛防护](#)。

故障自愈，在安全的前提下完成告警的自动化处理。

## Zabbix 告警自动处理

### 情景

故障处理是运维的职能之一，[Zabbix](#)自带[ActionScript](#)虽然可以实现告警自动处理，但存在2个问题：[无法集中管理自动处理的脚本](#)、[没有收敛防护，安全性无法保障](#)。

接下来我们通过将[Zabbix 中磁盘使用率 \(vfs.fs.\\*\) 告警接入故障自愈](#)这个案例，来了解故障自愈是如何解决这2个痛点。

### 前提条件

- 蓝鲸配置平台纳管了 Zabbix 监控的对象
- 拥有 Zabbix 管理员账号，用于注册 Zabbix Action

**术语解释 - 自愈套餐**：告警的处理动作，等同于 Zabbix 的 Action；**- 自愈方案**：关联告警和处理动作的一个组合；

### 操作步骤

- 接入 Zabbix 告警源
- 接入自愈方案
- 自愈测试

## 视频教程

{% video %}media/zabbix\_fta.mp4{% endvideo %}

### 接入 Zabbix 告警源

在菜单 [接入自愈] -> [管理告警源] 中，点击 启用 Zabbix。

The screenshot shows the BlueKing interface for managing alert sources. On the left sidebar, under the '接入自愈' (Integrate Self-recovery) section, there is a '管理告警源' (Manage Alert Sources) button. The main content area is titled '管理告警源' and includes a sub-section '已启用监控产品' (Enabled Monitoring Products). It lists several products: REST API (marked as '已禁用' - disabled), 蓝鲸监控 (BlueKing Monitoring), and Nagios. Below this is a section '未启用监控产品' (Disabled Monitoring Products), which includes ZABBIX (highlighted with a red box), falcon, Icinga 2, Amazon web services, and EMAIL. Each product entry has a '启用' (Enable) button.

[跳转到接入流程页面](#)



Zabbix 接入故障自愈的逻辑是，告警产生时，执行 `Action`，将告警推送至故障自愈接收告警的回调接口。

接下来，我们下载并初始化该 `Action`。

## 下载初始化脚本

参照上图，进入 Zabbix Action 的目录 `/usr/lib/zabbix/alertscripts`，下载初始化脚本

`zabbix_fta_alarm.py`。

```
[root@37ae504b6646 alertscripts]# wget 'http://${PaaS_Host}/o/bk_fta_solutions/0/alarm_source/scripts/zabbix_fta_alarm.py?fta_application_id=66fdfe50-3075-49bf-8101-d97386030c9b&fta_application_secret=EfgBbXD25N6870j9nkgf3ns8eOEsH2Sk' -O /usr/lib/zabbix/alertscripts/zabbix_fta_alarm.py --no-check-certificate
```

注：请直接复制故障自愈页面的命令，其中包含故障自愈的页面 URL 以及账号信息。

## 初始化 ZABBIX 告警配置

执行初始化 Zabbix 告警配置脚本 `zabbix_fta_alarm.py`，参数依次为 `--init`、`Zabbix API URL`、`Zabbix账号`、`Zabbix密码`

```
[root@37ae504b6646 alertscripts]# chmod +x zabbix_fta_alarm.py
[root@37ae504b6646 alertscripts]# ./zabbix_fta_alarm.py --init http://${Zabbix_Host}/api_jsonrpc.php Admin zabbix
[2019-07-30 10:51:45,374] INFO fta: get auth token: 136b14f3b8fe226bc02bc5eb4dfd7ac6
[2019-07-30 10:51:45,455] INFO fta: action_get success: {u'jsonrpc': u'2.0', u'result': [{u'actionid': u'8'}], u'id': 1}
[2019-07-30 10:51:45,572] INFO fta: action_delete success: {u'jsonrpc': u'2.0', u'result': {u'actionids': [u'8']}, u'id': 1}
[2019-07-30 10:51:45,640] INFO fta: user_get success: {u'jsonrpc': u'2.0', u'result': [
```

```

{u'userid': u'6'}], u'id': 1}
[2019-07-30 10:51:45,809] INFO fta: user_delete success: {u'jsonrpc': u'2.0', u'result': {u'userids': [u'6'], u'id': 1}}
[2019-07-30 10:51:45,902] INFO fta: mediatype_get success: {u'jsonrpc': u'2.0', u'result': [{u'mediatypeid': u'7'}], u'id': 1}
[2019-07-30 10:51:45,984] INFO fta: mediatype_delete success: {u'jsonrpc': u'2.0', u'result': {u'mediatypeids': [u'7']}, u'id': 1}
[2019-07-30 10:51:46,077] INFO fta: mediatype_create success: {u'jsonrpc': u'2.0', u'result': {u'mediatypeids': [u'8']}, u'id': 1}
[2019-07-30 10:51:46,174] INFO fta: user_create success: {u'jsonrpc': u'2.0', u'result': {u'userids': [u'7']}, u'id': 1}
[2019-07-30 10:51:46,274] INFO fta: action_create success: {u'jsonrpc': u'2.0', u'result': {u'actionids': [9]}, u'id': 1}

```

该脚本会创建一个名为 **FTA\_Event\_Handler** 的 Media Type, 名为 **FTA\_Act** 的 Action, 名为 **FTA\_Mgr** 的用户。

Name	Type	Status	Used in actions	Details	Action
Email	Email	Enabled		SMTP server: "smtp.163.com", SMTP helo: "smtp.163.com", SMTP email: "bluekingtest@163.com"	<a href="#">Test</a>
<b>FTA_Event_Handler</b>	Script	Enabled		Script name: "zabbix_fta_alarm.py"	<a href="#">Test</a>
Jabber	Jabber	Enabled		Jabber identifier: "jabber@example.com"	<a href="#">Test</a>
SMS	SMS	Enabled		GSM modem: "idevitySD"	<a href="#">Test</a>

## 接入自愈方案

Zabbix 告警源 接入成功后, 接下来关联告警 和 告警的处理动作。

将 Zabbix 中磁盘容量告警关联一个磁盘清理的处理动作。

选择菜单 [接入自愈] -> [接入自愈], 点击接入自愈

告警类型	告警数量	平台	集群	模块	自愈套餐	告警源	自愈方案	方案来源	启用	操作
REST默认分类	10	(所有)	(所有)	(所有)	<a href="#">默认测试套餐</a>	REST API监控	REST API测试	人工配置	<input checked="" type="checkbox"/>	<a href="#">编辑</a> <a href="#">删除</a>
磁盘容量(vfs.fs.*)	2	(所有)	(所有)	接入层	<a href="#">接入层日志...</a>	Zabbix监控	接入层日志...	人工配置	<input checked="" type="checkbox"/>	<a href="#">编辑</a> <a href="#">删除</a>
5分钟平均负载	0	(所有)	(所有)	(所有)	<a href="#">[快速] 发...</a>	蓝鲸监控	5分钟平均负载...	系统推荐	<input type="checkbox"/>	<a href="#">编辑</a> <a href="#">删除</a>
CPU单核使用率	0	(所有)	(所有)	(所有)	<a href="#">[快速] 发...</a>	蓝鲸监控	CPU单核使...	系统推荐	<input type="checkbox"/>	<a href="#">编辑</a> <a href="#">删除</a>
CPU总使用率	0	(所有)	(所有)	(所有)	<a href="#">[快速] 发...</a>	蓝鲸监控	CPU总使用...	系统推荐	<input type="checkbox"/>	<a href="#">编辑</a> <a href="#">删除</a>
交换分区使用量	0	(所有)	(所有)	(所有)	<a href="#">[快速] 发...</a>	蓝鲸监控	交换分区使...	系统推荐	<input type="checkbox"/>	<a href="#">编辑</a> <a href="#">删除</a>
可用物理内存	0	(所有)	(所有)	(所有)	<a href="#">[快速] 发...</a>	蓝鲸监控	可用物理内...	系统推荐	<input type="checkbox"/>	<a href="#">编辑</a> <a href="#">删除</a>
应用内存使用率	0	(所有)	(所有)	(所有)	<a href="#">[快速] 发...</a>	蓝鲸监控	应用内存使...	系统推荐	<input type="checkbox"/>	<a href="#">编辑</a> <a href="#">删除</a>
应用内存使用量	0	(所有)	(所有)	(所有)	<a href="#">[快速] 发...</a>	蓝鲸监控	应用内存使...	系统推荐	<input type="checkbox"/>	<a href="#">编辑</a> <a href="#">删除</a>
物理内存使用率	0	(所有)	(所有)	(所有)	<a href="#">[快速] 发...</a>	蓝鲸监控	物理内存使...	系统推荐	<input type="checkbox"/>	<a href="#">编辑</a> <a href="#">删除</a>
物理内存使用量	0	(所有)	(所有)	(所有)	<a href="#">[快速] 发...</a>	蓝鲸监控	物理内存使...	系统推荐	<input type="checkbox"/>	<a href="#">编辑</a> <a href="#">删除</a>

进入接入自愈页面, 告警类型选择 **磁盘容量(vfs.fs.\*)**, 自愈套餐点击 **+号** 新建 磁盘清理自愈

套餐。

The screenshot shows the 'Fault Recovery' interface. In the top navigation bar, there is a dropdown for 'Game (demo)', a bell icon for notifications, a 'Help' link, and a user profile for 'blueking'. On the left sidebar, there are several menu items: 首页 (Home), 自愈详情 (Recovery Details), 接入自愈 (Access Recovery), 入接入自愈 (Access Recovery), 套餐管理 (Plan Management), 管理告警源 (Manage Alert Sources), 高级配置 (Advanced Configuration), 统计报表 (Statistical Reports), and 管理员 (Administrator). The main content area is titled 'Recovery Scenario'. It includes sections for '告警类型' (Alert Type) set to '磁盘容量(vfs.fs.\*.)', '1. 将什么告警做自动化处理?' (1. What alerts will be automated handling?), '按内容筛选' (Filter by content), '平台' (Platform) set to '默认全选' (All selected), '集群' (Cluster) set to '默认全选' (All selected), and '模块' (Module) set to '接入层' (Access Layer). Below these are sections for '自愈处理' (Recovery Processing) and '自愈套餐' (Recovery Plan). The '自愈套餐' section contains fields for '选择处理告警的自愈套餐' (Select recovery plan for handling alerts), '查看' (View), and a '+' button. It also lists notification methods: '开始时' (At start) with checked boxes for WeChat, Email, and Phone; '成功时' (On success) with checked boxes for WeChat, Email, and Phone; and '失败时' (On failure) with checked boxes for WeChat, Email, and Phone. A note at the bottom says: '注：需要提前在蓝鲸控制台的【用户管理】填写联系方式' (Note: You need to fill in contact information in the [User Management] under the BlueKing Control Panel). There is also a note about creating recovery actions.

跳转至磁盘清理的自愈套餐页面。

The screenshot shows the 'Fault Recovery' interface. The left sidebar has the same menu items as the previous screenshot. The main content area is titled 'Recovery Plan'. It shows a table with two rows: '套餐类型' (Plan Type) set to '磁盘清理 (适用于Linux)' (Disk Cleaning (Applicable to Linux)) and '套餐命名\*' (Plan Name\*) set to '接入层日志清理' (Access Layer Log Cleaning). To the right of the table is a 'Recovery Plan Description' section with a note: '套餐是业务运维设计制作的一套恢复故障的方案，可以复用于不同的告警，也可作为原子套餐用于制作组合套餐。详情：【自愈套餐大全】' (The package is a set of recovery plans designed for fault recovery, which can be reused for different alerts, or used as atomic packages to build composite packages. Details: [Recovery Plan Encyclopedia]). Below the table, there are fields for '磁盘清理的目录' (Disk cleaning directory) set to '/data/logs/gse/', '删除多少天前的文件' (Delete files from how many days ago) set to '3天前' (3 days ago), and '待删除文件名描述' (Description of files to be deleted) set to '\*log'. At the bottom is a 'Save Recovery Plan' button.

保存自愈套餐后，自动回到接入自愈页面。添加完自愈方案后，在接入自愈列表页可以找到刚才创建的自愈方案。

The screenshot shows the 'Fault Recovery' interface. The left sidebar has the same menu items as the previous screenshots. The main content area is titled 'Access Recovery'. It displays a table of recovery plans. The first row is 'REST默认分类' (Default REST Category) with 10 alerts, all from '(所有)' (All) and '(所有)' (All), using '默认测试套餐' (Default Test Plan) via 'REST API监控' (REST API Monitoring) and '人工配置' (Manual Configuration) status '是' (Yes). The second row is '磁盘容量(vfs.fs.\*.)' (Disk Capacity) with 0 alerts, all from '(所有)' (All) and '(所有)' (All), using '接入层磁盘...' (Access Layer Disk...) via 'Zabbix监控' (Zabbix Monitoring) and '人工配置' (Manual Configuration) status '是' (Yes). The third row is '5分钟平均负载' (Average Load over 5 Minutes) with 0 alerts, all from '(所有)' (All) and '(所有)' (All), using '【快排】发...' (Fast Sort Triggered) via '蓝鲸监控' (BlueKing Monitoring) and '系统推荐' (System Recommendation) status '否' (No). The fourth row is 'CPU单核使用率' (Single-core CPU Usage Rate) with 0 alerts, all from '(所有)' (All) and '(所有)' (All), using '【快排】发...' (Fast Sort Triggered) via '蓝鲸监控' (BlueKing Monitoring) and '系统推荐' (System Recommendation) status '否' (No). The fifth row is 'CPU总使用率' (Total CPU Usage Rate) with 0 alerts, all from '(所有)' (All) and '(所有)' (All), using '蓝鲸监控' (BlueKing Monitoring) and '人工配置' (Manual Configuration) status '否' (No). There is a green button '接入自愈' (Access Recovery) at the top right of the table.

## 自愈测试

生成一个大文件，使磁盘剩余空间低于 20% ( Zabbix 中默认设定的 Trigger 是<20% )

```
[root@access_layer_breaking gse]# pwd
/data/logs/gse
[root@access_layer_breaking gse]# touch -d "10 days ago" agent-20190726-00001.log
[root@access_layer_breaking gse]# ll
总用量 3906980
-rw-r--r-- 1 root root 4000000000 7月 20 11:38 agent-20190726-00001.log
-rw-r--r-- 1 root root 196795 7月 26 23:59 agent-20190726-00002.log
-rw-r--r-- 1 root root 194952 7月 27 23:59 agent-20190727-00003.log
-rw-r--r-- 1 root root 198532 7月 28 23:59 agent-20190728-00004.log
-rw-r--r-- 1 root root 142948 7月 29 17:33 agent-20190729-00005.log
[root@access_layer_breaking gse]# dd if=/dev/zero of=4gb.log bs=1GB count=4
记录了4+0 的读入
记录了4+0 的写出
4000000000字节(4.0 GB)已复制, 34.0365 秒, 118 MB/秒
```

稍等片刻，收到 Zabbix 邮件告警，以及故障自愈的处理通知。



在故障自愈的 [自愈详情] 菜单中也可以找到自愈记录。

A screenshot of the Blueking fault recovery interface. The left sidebar shows navigation options like Home, Recovery Details, Connect Recovery, Advanced Configuration, Statistics Report, and Administrator. The main area is titled "故障自愈" and "欢乐游戏(demo)" with a date filter "2019-07-29 - 2019-07-29". It displays a table of recovery events. The table has columns: 类型 (Type), 产生时间 (Occurrence Time), 自愈耗时 (Recovery Duration), 平台 (Platform), 集群 (Cluster), 模块 (Module), IP, 状态 (Status), and 自愈结果 (Recovery Result). Two entries are shown: "磁盘容量(vfs.fs.\*) 07-29 20:07:34 +0800 23秒 Android\_Weixin 一区 接入层 10.0.4.128 成功 执行Job作业成功[337]" and "磁盘容量(vfs.fs.\*) 07-29 19:52:34 +0800 29秒 Android\_Weixin 一区 接入层 10.0.4.128 成功 执行Job作业成功[336]". A pagination bar at the bottom indicates page 1 of 1.

可以查看详情执行记录。

A screenshot of the fault recovery detail page for a specific event. At the top, it shows "告警详情" (Alert Details) for "业务 欢乐游戏(demo)" and "处理过程" (Handling Process) steps. The "处理状态" (Handling Status) is "成功 执行Job作业成功[337]". Below this, there's an "操作" (Operation) button labeled "重试整个流程" (Retry entire process). At the bottom, a green button labeled "#0 [43] 接入层日志清理" (Access Layer Log Cleaning) is visible.

从告警产生到处理结束，耗时 30 秒。

**告警自动处理，如此简单，不要再手动登录到服务器上处理告警了。**

## 扩展阅读

### 告警收敛确保安全的告警自动处理

选择菜单 **[高级配置] -> [告警收敛]**，新增高危告警的收敛规则，比如 Ping 告警。

一般网络波动时，可能触发假的 Ping 告警，这时不能直接重启服务器，可以通过告警收敛，让运维审批。

具体执行记录，请参照 [故障自愈的收敛防护](#)

## 健康度日报

如果服务器频繁触发自愈，那么需要去思考本质问题是什么，是磁盘使用率的告警策略不合适，还是主板故障，亦或是程序运行异常。

这时可以使用 预警功能。

我们在做该教程的时候，在同一天触发了 2 条磁盘使用率的自愈记录，所以产生了一条健康诊断记录。



The screenshot shows the BlueKing Health Diagnosis interface. On the left is a sidebar with navigation links: 首页 (Home), 自愈详情 (Recovery Details), 接入自愈 (Access Recovery), 高级配置 (Advanced Configuration) (selected), 告警收敛 (Alert Convergence), 告警详情 (Alert Details), 接入预警 (Access Alert), 自愈小助手 (Recovery Assistant), 统计报表 (Statistical Reports), and 管理员 (Administrator). The main content area is titled "健康诊断" (Health Diagnosis) with the subtitle "依据预警自愈配置的策略，每天早上8点回溯自愈处理过的告警，处理分析出的潜在风险。". It displays a table with one row of data:

告警类型	发现时间	发生频次	可疑IP	平台	集群	模块	处理类型	处理方案	状态	事件详情
磁盘容量(v...)	07-30	一天内2次	10.0.4.128	...	一区	接入层	建议	1. 检查磁盘容量 2. 确认该模块的硬盘空间是否合理	待处理	<a href="#">查看详情</a>

在邮件中也可以找到。



The email subject is "[自愈通知] 健康度日报 欢乐游戏(demo) 07月30天 1个新风险点 ☆". The header includes: 发件人: breaking <bluekingtest@163.com>, 时 间: 2019年7月30日(星期二) 上午8:00, 收件人: Pythoning <pythoning@qq.com>; Pythoning <pythoning@qq.com>. The body of the email features a large blue banner with the text "故障自愈 · 你的无人值守伙伴!" and a shield logo. Below the banner, the message starts with "早上好，亲爱的欢乐游戏(demo)运维帝：恭喜你，当你收到这封邮件，说明你的业务享受到了故障自愈的深度服务，以下是自愈今天最新的分析报告。". It then lists a new risk point: "以下是你今天自愈新发现的可疑故障点，建议在故障前尽早消除，降低业务损失的风险。猛击 [欢乐游戏\(demo\) 健康度完整报告](#)，查看所有的可疑故障点，全面整顿！". A red warning box highlights "host 10.0.4.128 一天内 2次 磁盘容量(vfs.fs.\*)" with the following details: 集群: 一区, 模块: 接入层, 建议: 1、请检查当前的磁盘清理策略是否需要调整  
2、确认该模块当前机型的硬盘空间是否合理.

可在菜单 [高级配置] -> [接入预警] 中设置。

## 第 3 方监控系统告警自动处理

### 情景

故障处理是运维的职能之一，人工登录服务器处理告警，存在 2 个问题：**故障处理效率低** 和 **操作疏忽时可能影响生产环境**，例如删除文件输入绝对路径时，在根目录和日志目录间误敲空格，导致根目录删除。

前面介绍了**蓝鲸监控**、**Zabbix** 告警的自动处理，接下来通过“**REST API 告警接入故障自愈**”这个案例，来了解故障自愈如何集成第 3 方监控系统。

## 前提条件

- 蓝鲸配置平台纳管了主机
- 作业平台新建一个作业

**术语解释 - 自愈套餐**：告警的处理动作，比如清理日志的作业 - **自愈方案**：关联告警和处理动作的一个组合

## 操作步骤

- 启用 REST API(推送)告警源
- 接入自愈方案
- 自愈测试
- 故障自愈的收敛防护

### 启用 REST API(推送)告警源

第 3 方监控系统调用 REST API(推送)接口，故障自愈收到告警后立即做自动化处理。



第3方监控系统 告警自动处理 流程图

首先，启用告警源。

在菜单 **[接入自愈] -> [管理告警源]** 中，启用 **REST API(推送)**。

Copyright © 2012-2019 Tencent  
BlueKing.  
All Rights Reserved.

## 接入自愈方案

在菜单 [接入自愈] 中，点击 **接入自愈**，告警类型选择 **REST默认分类**。

点击新建 **自愈套餐** 的按钮，准备一个告警的处理动作。

返回上一页

告警类型 **REST默认分类** 1. 将什么告警做成自动处理?

平台 默认全选 集群 默认全选

模块 默认全选

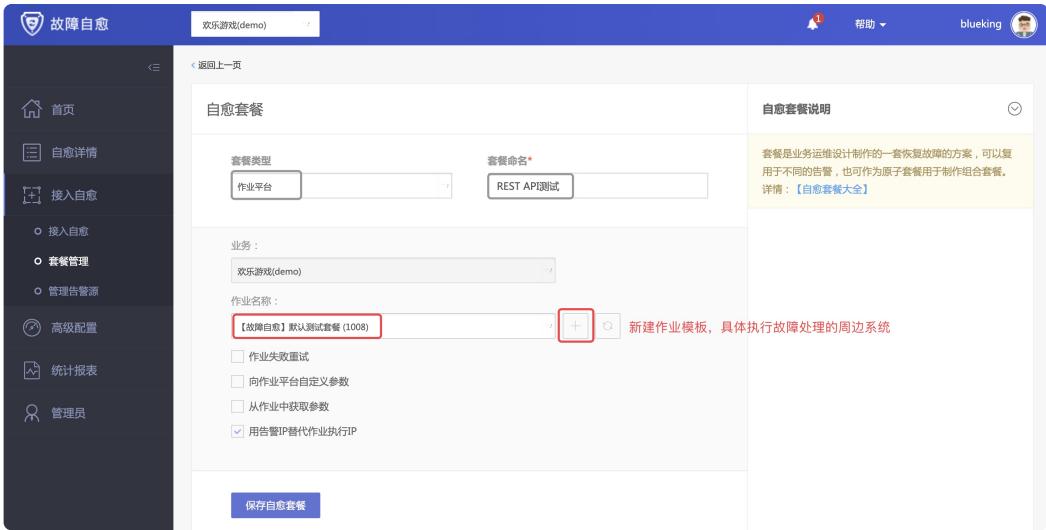
自愈套餐 请选择处理告警的自愈套餐 + 2. 告警处理的动作是什么?

通知方式  
开始时  微信  邮件  短信  电话  
成功时  微信  邮件  短信  电话  
失败时  微信  邮件  短信  电话

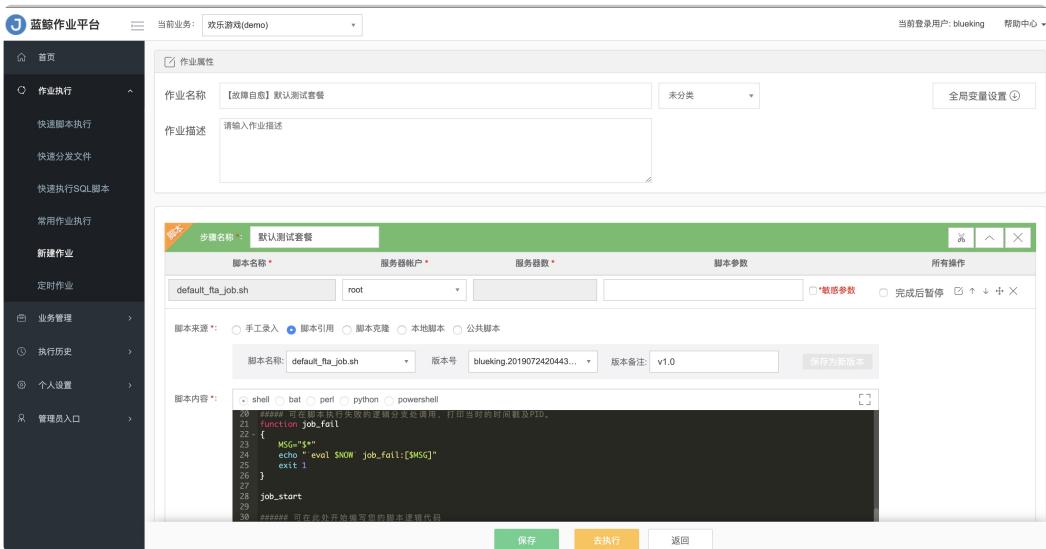
通知人员  业务运维: blueking.admin  
 更多通知人

注: 需要提前在蓝鲸控制台的【用户管理】填写联系方式

在自愈套餐页面，套餐类型选择 **作业平台**，点击作业名称右侧的加号，新建一个测试的作业模板。



点击 **新建作业的按钮** 后，跳转至作业平台，在菜单 **[作业执行] -> [新建作业]** 中，新建一个默认的作业即可。



保存 **REST API 测试** 的自愈套餐后，自动回到接入自愈的页面，保存自愈方案即可。

回到接入自愈列表，在列表中可以找到刚刚创建的自愈方案。

## 自愈测试

在 **REST API(推送)** 的告警源管理页面，复制调用实例。

将示例中的 IP 替换给该业务下任意一个 IP，然后贴到终端下执行。

6

如果多次调试，请保证 `source_id` 唯一（重复会丢弃），`source_time` 和服务器时间一致（过长会丢弃）。

”

在 [自愈详情](#) 页面可以找到自愈记录。

The screenshot shows the 'Fault Recovery' interface for the '欢乐游戏(demo)' application. The top navigation bar includes a shield icon, the application name, a search bar, a notification bell with one alert, a 'Help' dropdown, and a user profile for 'blueking'. On the left, a sidebar lists '首页', '自愈详情', '接入自愈', '高级配置', '统计报表', and '管理员'. The main content area has a title '返回上一页' and a date range selector from '2019-07-31' to '2019-07-31'. It displays a table with a single row of data:

类型	产生时间	自愈耗时	平台	集群	模块	IP	状态	自愈结果
REST默认分类	07-31 13:11:28 +0800	22秒	Android_Weixin	一区	存储层	10.0.4.29	<span>✓</span>	执行/job作业成功[3/1]

At the bottom right of the table, there are navigation buttons for page 1 of 1.

点开 状态 按钮，可以查看详情。

告警详情	业务 欢乐游戏(demo) 的主机 10.0.4.29 在 2019-07-31 13:11:28 +0800 发生 REST默认分类 : FAILURE for production HTTP on machine 10.0.0.1
处理过程	13:14:19 开始处理套餐[REST API测试] 13:14:39 #0 REST API测试   成功: 执行Job作业成功[371]
处理状态	<span>成功</span> 执行Job作业成功[371]
操作	<span>重试整个流程</span>

点击 [详情](#) 中的作业执行 ID，可查看执行的作业。

至此，一次模拟告警的故障自愈演示完毕。

REST API(推动)的场景在于，如果你使用的监控系统故障自愈默认未集成，则可以通过回调 REST API 的方式，将告警推送至故障自愈，故障自愈执行对应的处理动作，完成告警的自动处理。

## 故障自愈的收敛防护

故障自动处理是把双刃剑，需要考虑因为网络波动等场景导致的假告警，这时可以用到故障自愈的 **异常防御需审批** 功能。

在菜单 **高级配置 -> 告警收敛**，新建 1 条收敛规则。

告警类型选择 **REST默认分类**，条件为 **相同业务**，触发频次为 **5分钟2次以上**，收敛方式为 **异常防御需审批**，备注填写 **疑似网络波动，请审批**。

执行 4 次告警推送后，可以看到多条告警收敛为 1 条告警，需要运维审批。

The screenshots illustrate the 'Fault Recovery' feature in the BlueKing ITSM platform. The top screenshot shows a single recovery event for a network fluctuation, with a note to review the log. The bottom screenshot shows multiple recovery events, including REST API and Job execution logs.

故障自愈，在安全的前提下完成告警的自动化处理。

“

收敛审批是通过企业微信实现，请参考 [微信审批接入流程](#)。

”

## 服务器资源申请流程线上化

### 情景

传统的资源申请是 **通过邮件**，沟通成本太高，而且无法回溯。

新业务上线，需要申请几台服务器用于部署应用。接下来，看蓝鲸 ITSM 是如何 实现服务器申请的流程线上化。

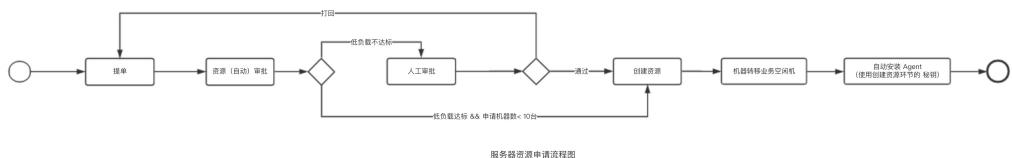
### 前提条件

- 蓝鲸企业版，自带 ITSM SaaS。
- 准备 1 个管理员账号，用于流程设计；1 个运维账号用于**资源申请**，1 个**普通账号 SA** 用于**审批和交付资源**。

## 操作步骤

- 梳理资源申请流程
- 创建资源申请服务目录及流程
- 一次资源申请示例

### 梳理资源申请流程



“  
流程图中是一个实践案例，部分数据需要从周边系统获取，此处功能需要做二次开发，本教程专注流程本身。  
”

### 创建资源申请服务目录及流程

先设计资源申请的流程，流程依附在服务目录上对外提供服务。

### 角色设置

使用蓝鲸管理员账号给该业务分配运维人员的角色。

该界面展示了蓝鲸配置平台的业务管理模块。左侧是导航栏，包含首页、基础资源、业务、主机、业务资源、审计与分析、权限控制、模型管理、我的收藏等。右侧是“业务”管理页面，显示了“新建”按钮和一个名为“欢乐游戏(demo)”的业务记录。该记录包括ID、业务名、运维人员等信息。右侧还显示了“属性”、“关联”、“变更记录”、“基础信息”（包含业务名、生命周期、时区、语言）、“角色”（包含运维人员、测试人员、操作人员、产品人员、开发人员）等详细信息。底部有“编辑”和“归档”按钮。

使用管理员账号新增通用角色 SA，并添加用户 sa\_zhang（该账户是前提条件下准备的）。

No.	角色名	人员	操作
1	服务台小组	admin(管理员)	<button>编辑</button> <button>删除</button>
2	故障源单员	admin(管理员)	<button>编辑</button> <button>删除</button>
3	变更经理	admin(管理员)	<button>编辑</button> <button>删除</button>
4	测试		<button>编辑</button> <button>删除</button>
5	运维		<button>编辑</button> <button>删除</button>
6	运营		<button>编辑</button> <button>删除</button>
7	开发经理		<button>编辑</button> <button>删除</button>
8	开发		<button>编辑</button> <button>删除</button>

## 设计资源申请流程

### 填写流程信息

选择菜单【流程设计】，点击【新增】按钮，按提示填写流程信息。

流程名称：\* 申请服务器

流程说明：\* 业务上线、扩容、变更等场景需要的计算资源。

流程类型：\* 请求

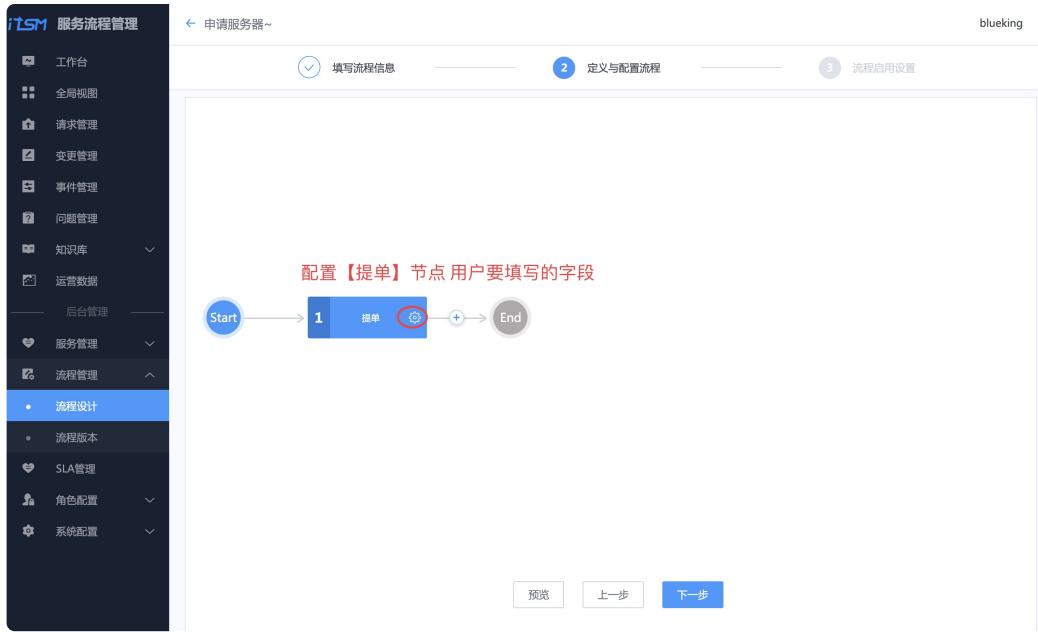
是否关联业务：\*  是  否

资源申请类，一般选择“请求”

流程类型选择【请求】，需要【关联业务】，因为资源申请和业务相关，资源管理员需要知道将资源分配给哪个业务。

点击【下一步】，进入【定义与配置流程】环节。

### 定义与配置流程



点击上图【流程画布】中的【齿轮】，配置【提单】流程节点的字段。

一般是业务的运维提服务器资源申请的单据，所以操作角色选择【CMDB 业务公用角色】->【运维人员】。

**基本信息**

节点名称:	提单	一般是 运维 申请服务器					
操作角色:	CMDB业务公用角色	运维人员					
<b>+ 新增字段</b>							
字段名	字段类型	字段值	字段描述	字段校验方式	布局要求	字段属性	操作
标题	单行文本	--	申请N台服务器	必填	半行	自定义	上移 下移 编辑
用途	单选下拉框	业务上线,业务扩容	申请服务器做什么	必填	整行	自定义	上移 下移 编辑 删除
期望交付时间	日期	--	期望交付时间	必填	整行	自定义	上移 下移 编辑 删除
服务器配置	自定义表格	--	各种机型需要多少	可选	整行	自定义	上移 下移 编辑 删除

**保存** **字段预览**

点击【新增字段】，参照 梳理资源申请流程 ，新增每个环节中需要的字段。

**基本信息**

字段名	字段类型	字段值	字段描述
标题	单行文本	--	申请N台服务器
用途	单选下拉框	业务上线,业务扩容	申请服务器做什么
期望交付时间	日期	--	期望交付时间
服务器配置	自定义表格	--	各种机型需要多少

**编辑字段**

字段显示名：

用途：

字段描述：

申请服务器做什么  
至少输入5个字 8/100

字段类型：  
 单选下拉框 **选择合适的字段类型**

数据源：  
 自定义数据  接口数据  数据字典

自定义数据：  
业务上线  
业务扩容

布局要求：  
整行

字段校验方式：  
必填

正则规则表达式：  
无

**提交** **取消**

参照 **梳理资源申请流程**，完成整个服务器申请流程的配置。



### 启用流程

**【启用流程】**，选择适合的通知策略，点击**【提交】**完成流程设计。

**设置流程启用管理**

是否启用该流程： ON

**设置任务通知策略**

是否通知： 是  否

是否督办： 是  否

通知频率： 首次通知，以后不再通知  
 首次通知后，次日起每天定时通知

通知类型： SMS短信  邮件  微信

[上一步](#) [提交](#)

### 流程模板实例化

选择菜单【流程设计】，找到刚编辑的申请服务器流程，点击【部署】，生成流程实例。

**流程设计**

No.	流程名	说明	类型	更新人	更新时间	状态	操作
1	应用发布	将应用版本发布至生产环境	变更	blueking(blueking)	2019-08-14 16:15:25	启用	预览 编辑 导出 部署 删除
2	申请服务器	业务上线、扩容、变更等场景需要的计算资源。	请求	blueking(blueking)	2019-08-14 20:06:01	启用	预览 编辑 导出 部署 <input checked="" type="checkbox"/> 删除

共计1页

确认部署此流程？

流程部署成功以后，会产生对应的流程版本

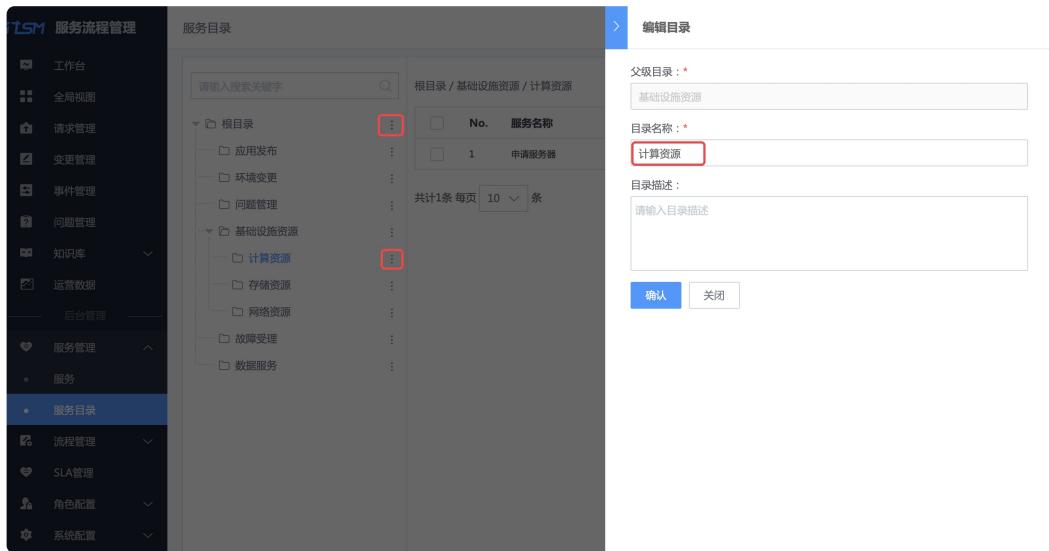
[确定](#) [取消](#)

### 在服务目录中新增 "申请服务器" 服务，并绑定流程

选择菜单【服务】，点击【新增】按钮，新增"申请服务器"服务，并关联刚生成的流程实例。



选择菜单【服务目录】，选中【根目录】，点击右侧【：】，点击【新增】，按提示新增一个名为基础设施资源的服务目录，然后在基础设施资源下新建一个计算资源的服务目录。

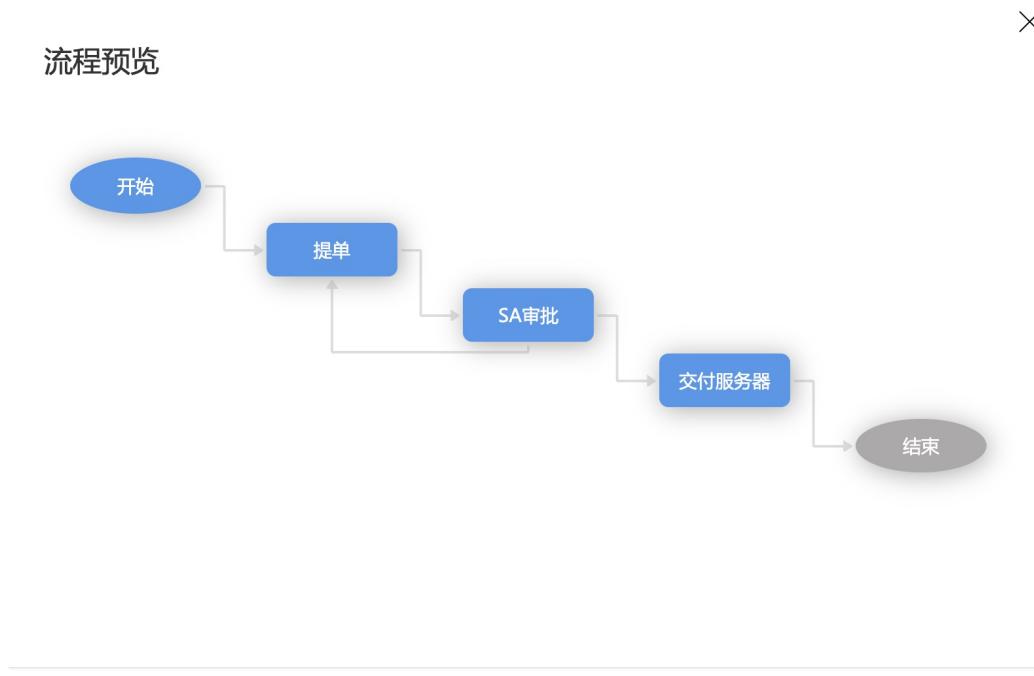


选中刚刚创建的服务目录【计算资源】，右侧会显示【添加】按钮，点击该按钮添加申请服务器服务。

The screenshot shows the ITSM Service Catalog interface. On the left is a sidebar with various management modules like Workstation, Full View, Request Management, Change Management, Event Management, Problem Management, Knowledge Base, Operation Data, Backend Management, Service Management, and Services. The 'Service Catalog' option is selected. The main area has a search bar and a tree view of service categories: Root Catalog, Application Release, Environment Change, Problem Management, Infrastructure Resources (Compute Resources, Storage Resources, Network Resources, Fault Management, Data Services), and a summary of 0 items per page. A modal window titled 'Add Service' is open, showing a table with one row: No. 1, Service Name '申请服务器', and Service Type '请求管理'. Buttons for 'Confirm' and 'Close' are at the bottom.

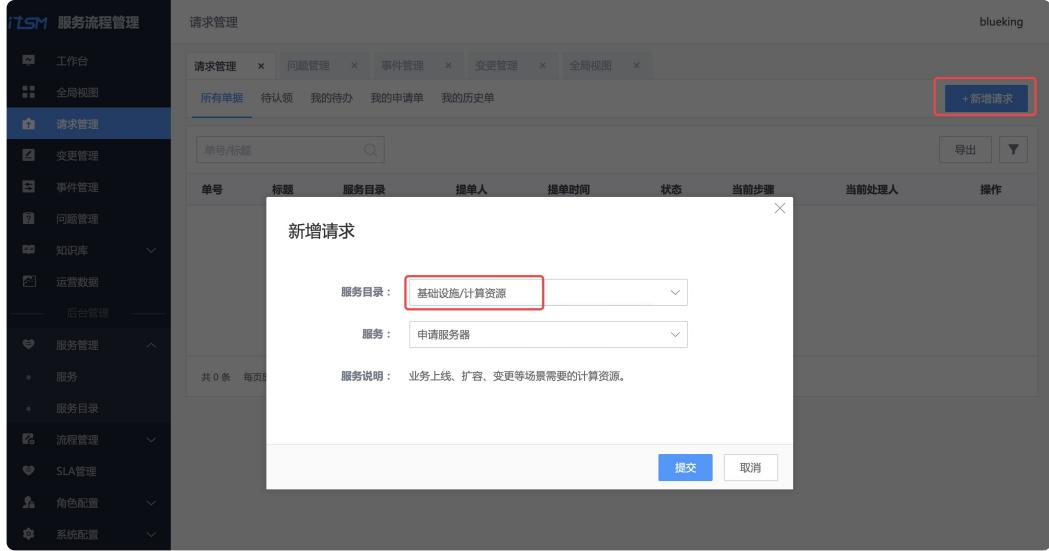
## 一次资源申请示例

创建完“申请服务器”流程及服务目录后，接下来做一次服务器申请演示。



## 运维提单申请服务器

用[运维](#)账号登录ITSM，选择【请求管理】菜单，点击【新增请求】，选择【申请服务器】服务，点击【提交】。



在申请服务器的提单界面，填写服务器申请的关键信息：关联业务、期望交付时间以及服务器的配置和地域。为了安全起见，逻辑层和存储层一般不需要外网 IP。

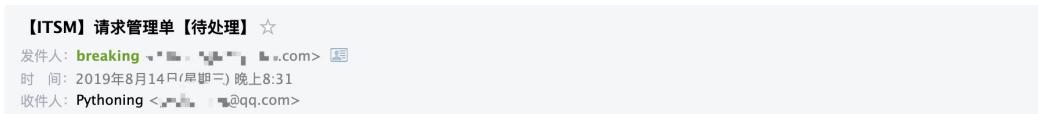
点击【提交】，创建服务器申请需求。

机型	操作系统	外网IP	可用区	数量	操作
A1(计算):8核/16G/1T/SAS/NORaid	CentOS 7.6	是	广州	2	[Add] [Delete]
B1(内存):8倍/64G/1T/SAS/RAID1	CentOS 7.6	否	广州	2	[Add] [Delete]
C2(存储):8核/16G/1T/SAS/RAID1	CentOS 7.6	否	广州	2	[Add] [Delete]

注：点击【保存模板】可以记录提单环节的关键信息，下次提单只需修改少部分字段即可。

## SA 审批单据

SA 收到一封待处理的服务器申请邮件。



### 【ITSM通知】请求管理

Hi,您好!

您有1条请求管理单 【待处理】

标题: 欢乐游戏业务扩容, 申请6台服务器

单号: NO2019081420252287

服务目录: 基础设施资源->计算资源

当前环节: SA审批

当前处理人: sa\_zhang(SA)

[点击查看详情](#)

如有任何问题, 可随时联系ITSM助手。

ITSM流程管理服务

2019年8月14日

使用 SA 账号登录 ITSM, 在待办列表中, 找到一个刚刚运维提交的服务器申请单据。



点击链接, 同时在周边系统查看该业务的低负载指标, 确认达标, 点击【通过】, 完成本环节流程。

ITSM 服务流程管理

NO201908142025287 标题: 欢乐游戏业务扩容,申请6台服...

工单处理

**提单** 处理人: blueking(blueking) 处理操作: 提单

标题: 欢乐游戏业务扩容,申请6台服务器  
用途: 业务扩容  
服务器配置:

机型	操作系统	外网IP	可用区	数量
A1(计算)8核/16G/1T/SAS/NORAD	CentOS 7.6	是	广州	2
B1(内存)8核/64G/1T/SAS/RAID1	CentOS 7.6	否	广州	2
C2(存储)8核/16G/1T/SSD/RAID1	CentOS 7.6	否	广州	2

处理时间: 2019-08-14 20:31:38

**SA审批** 低负载指标 达标, 通过资源审批

通过 打印 保存字段

流转日志 通知关注人  
2019-08-14 20:31:38 blueking(blueking) 处理节点【提单】(提交)

“

低负载指标: CPU、内存、网络等指标是否长期处于一个很低的水平, 如果是的话, 证明计算资源利用率低, SA一般不予通过资源申请。

”

## 交付服务器

SA创建资源, 并录入到【交付服务器】环节的服务器配置中。

ITSM 服务流程管理

NO201908142025287 标题: 欢乐游戏业务扩容,申请6台服...

用途: 业务扩容  
服务器配置:

机型	操作系统	外网IP	可用区	数量
A1(计算)8核/16G/1T/SAS/NORAD	CentOS 7.6	是	广州	2
B1(内存)8核/64G/1T/SAS/RAID1	CentOS 7.6	否	广州	2
C2(存储)8核/16G/1T/SSD/RAID1	CentOS 7.6	否	广州	2

处理时间: 2019-08-14 20:31:38

**SA审批** 处理人: sa\_zhang(sa\_zhang) 处理操作: 通过

**交付服务器**

服务器配置:

内网IP	外网IP	机型	操作
10.0.4.21	123.2.1.11	A1(计算)8核/16G/1T/SAS/NORAD	添加 删除
10.0.4.22	123.2.1.21	A1(计算)8核/16G/1T/SAS/NORAD	添加 删除
10.0.4.23		B1(内存)8核/64G/1T/SAS/RAID1	添加 删除
10.0.4.24		B1(内存)8核/64G/1T/SAS/RAID1	添加 删除
10.0.4.25		C2(存储)8核/16G/1T/SSD/RAID1	添加 删除
10.0.4.26		C2(存储)8核/16G/1T/SSD/RAID1	添加 删除

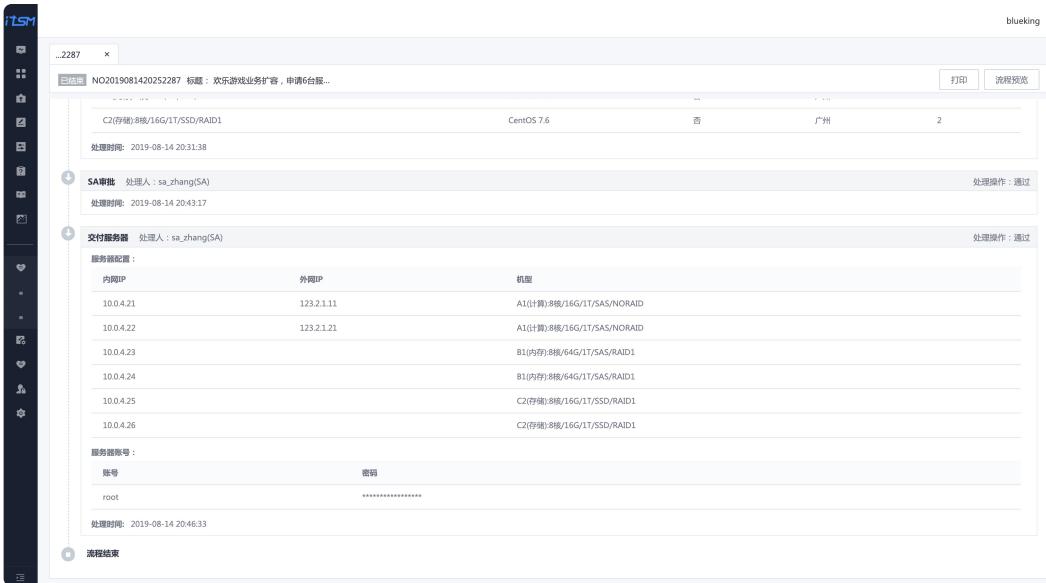
服务器账号: \*

账号	密码	操作
root	*****	添加 删除

通过 保存字段

至此, 服务器申请流程结束。

运维登录 ITSM, 即可查看申请的服务器资源。



# 发布流程线上化

## 情景

应用发布是运维的职能之一，传统的应用发布通过邮件交付，沟通成本太高，而且不符合合规性检查。

接下来，以一个业务的应用发布为例，看蓝鲸 ITSM 是如何解决这个痛点的。

## 前提条件

- 蓝鲸企业版，自带 ITSM SaaS。
- 准备发布流程中的 **各个角色** 的账号，包含流程设计的 **管理员**，以及应用发布流程中的 **产品**、**运维**、**测试**、**QC**。

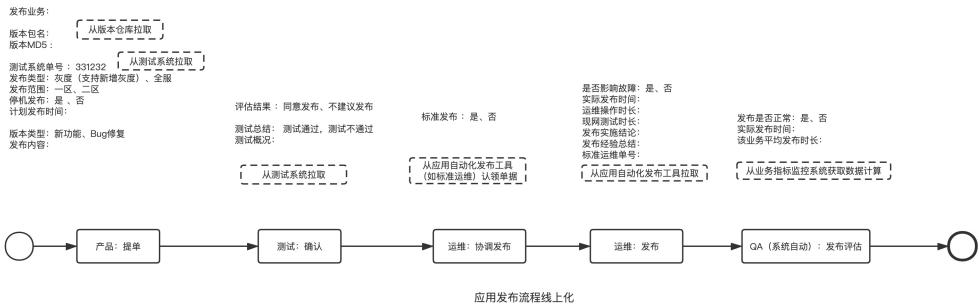
## 操作步骤

- 梳理应用发布流程
- 创建应用发布服务目录及流程
- 一次应用发布示例

### 梳理应用发布流程

从 ITSM 理论出发，梳理一个应用发布流程图，将测试环境验收通过的版本，部署到生产环境。包含研发运营环节的测试、产品提交发布需求、运维协调发布资源、发布，以及最后质量保证

## (QA)对应用发布质量的管理。



“

流程图中是一个实践案例，部分数据需要从周边系统获取，此处功能需要做二次开发，本教程专注流程本身。

”

## 创建“应用发布”流程及服务目录

先设计应用发布的流程，流程依附在服务目录上对外提供服务。

### 角色设置

使用蓝鲸管理员账号给该业务分配对应角色的权限。

The screenshot shows the BlueKing Configuration Platform interface. On the left, the sidebar has '业务' (Business) selected under '基础资源'. The main area shows a table for the '欢乐游戏(demo)' service, with columns for ID, 业务名 (Service Name), and 运维人员 (Operations Personnel). The table lists several entries, including '蓝鲸(blueking)' and '欢乐游戏(demo)'. On the right, the '属性' (Properties) tab is selected for the '欢乐游戏(demo)' service. Under the '基本信息' (Basic Information) section, it shows the service name, life cycle (已上线), time zone (Asia/Shanghai), and language (Chinese). In the '角色' (Roles) section, there are four roles listed: '运维人员' (blueking(blueking), ops\_a(运维小A)), '产品人员' (chanpin(产品)), '测试人员' (ceshi(测试)), and '操作人员' (caozuo(操作A)). The '运维人员' role is highlighted with a red box. At the bottom, there are '编辑' (Edit) and '归档' (Archive) buttons.

使用管理员账号新增通用角色 QC，并添加用户 qc\_c（该账户是前提条件中准备的）。

No.	角色名	人员	操作
1	QC	qc_c(QC/jC)	<a href="#">编辑</a> <a href="#">删除</a>
2	SA	sa_zhang(SA)	<a href="#">编辑</a> <a href="#">删除</a>
3	服务台小组	admin(管理员)	<a href="#">编辑</a> <a href="#">删除</a>
4	故障派单员	admin(管理员)	<a href="#">编辑</a> <a href="#">删除</a>
5	变更经理	admin(管理员)	<a href="#">编辑</a> <a href="#">删除</a>
6	运营	admin(管理员),sa_zhang(SA)	<a href="#">编辑</a> <a href="#">删除</a>

## 设计应用发布流程

### 填写流程信息

选择菜单【流程设计】，点击【新增】按钮，按提示填写流程信息。

填写流程信息

流程名称：\* 应用发布

流程说明：\* 将应用版本发布至生产环境

流程类型：\* 变更

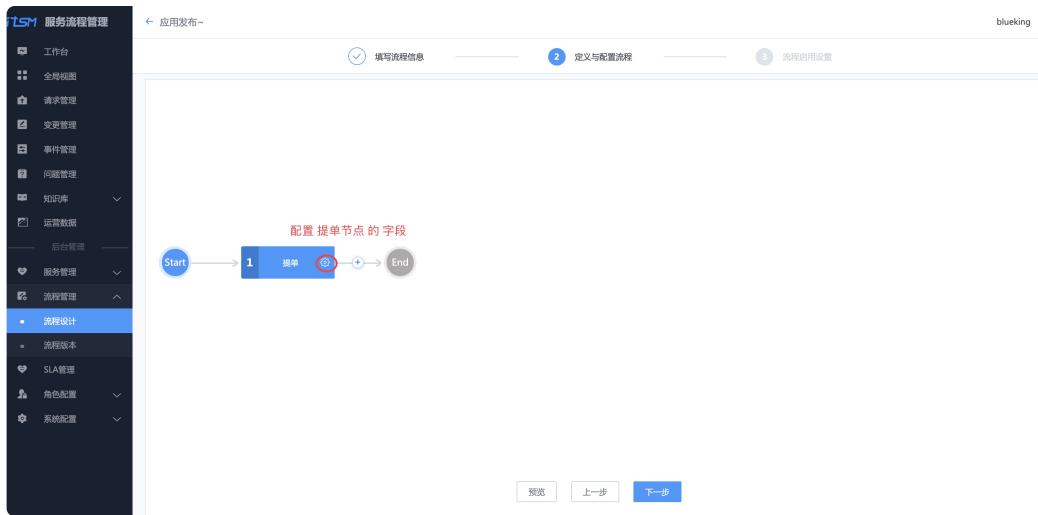
是否关联业务：  是  否

返回 下一步

流程类型选择【变更】，需要【关联业务】，因为应用发布和业务相关，同时关联业务对应的角色：产品、运维、测试。

点击【下一步】，进入【定义与配置流程】环节。

### 定义与配置流程

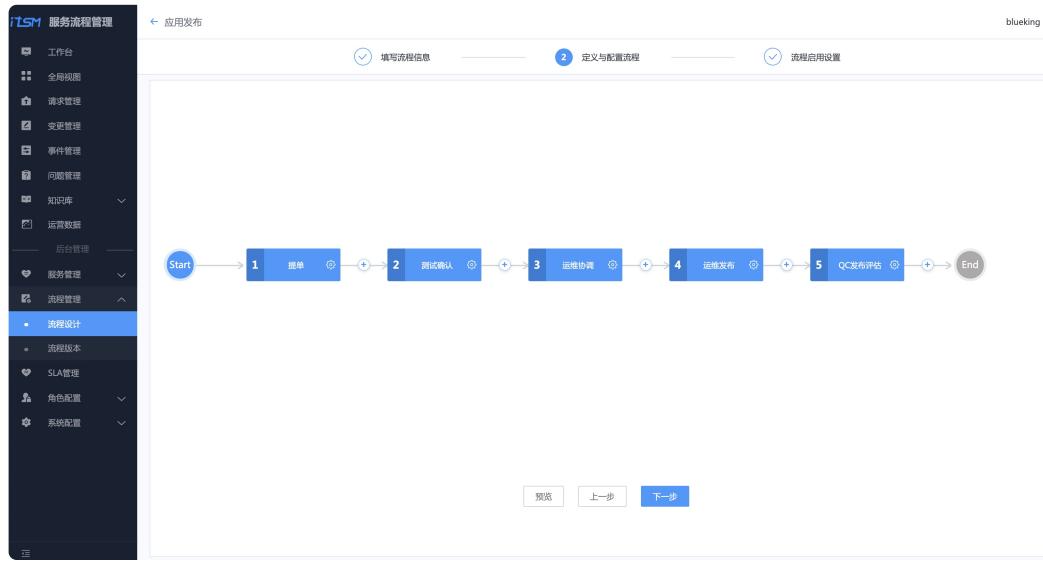


点击上图【流程画布】中的【齿轮】，配置【提单】流程节点的字段。

一般是业务的产品提应用的发布单据，所以操作角色选择【CMDB 业务公用角色】->【产品人员】。

点击【新增字段】，参照梳理应用发布流程，新增每个环节中需要的字段。

参照 **梳理应用发布流程**，完成整个应用发布流程的配置。



## 启用流程

【启用流程】，选择适合的通知策略，点击【提交】完成流程设计。

设置流程启用管理  
是否启用该流程：  ON

设置任务通知策略  
是否通知： 是  否  
是否督办： 是  否  
通知频率： 首次通知, 以后不再通知  首次通知后, 次日起每天定时通知  
通知类型： SMS短信  邮件  微信

## 流程模板实例化

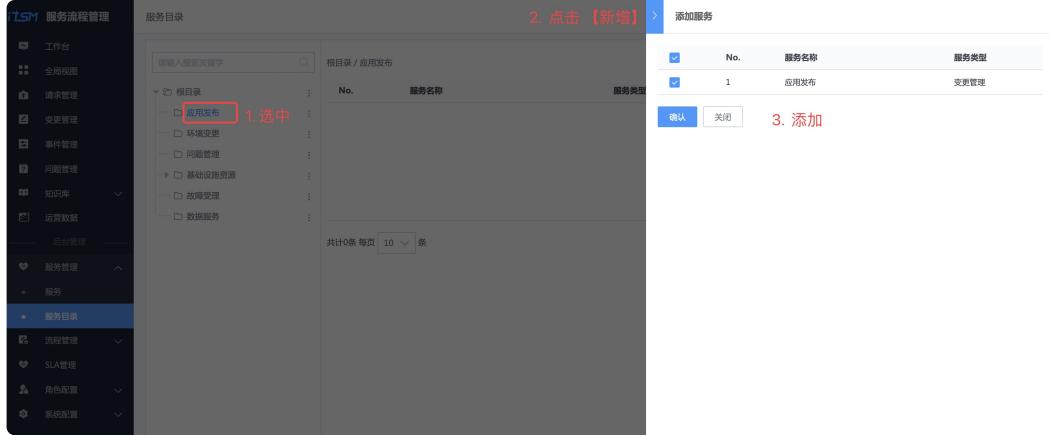
选择菜单【流程设计】，找到刚编辑的应用发布流程，点击【部署】，生成流程实例。

## 在服务目录中新增"应用发布"服务，并绑定流程

选择菜单【服务】，点击【新增】按钮，新增“应用发布”服务，并关联刚生成的流程实例。

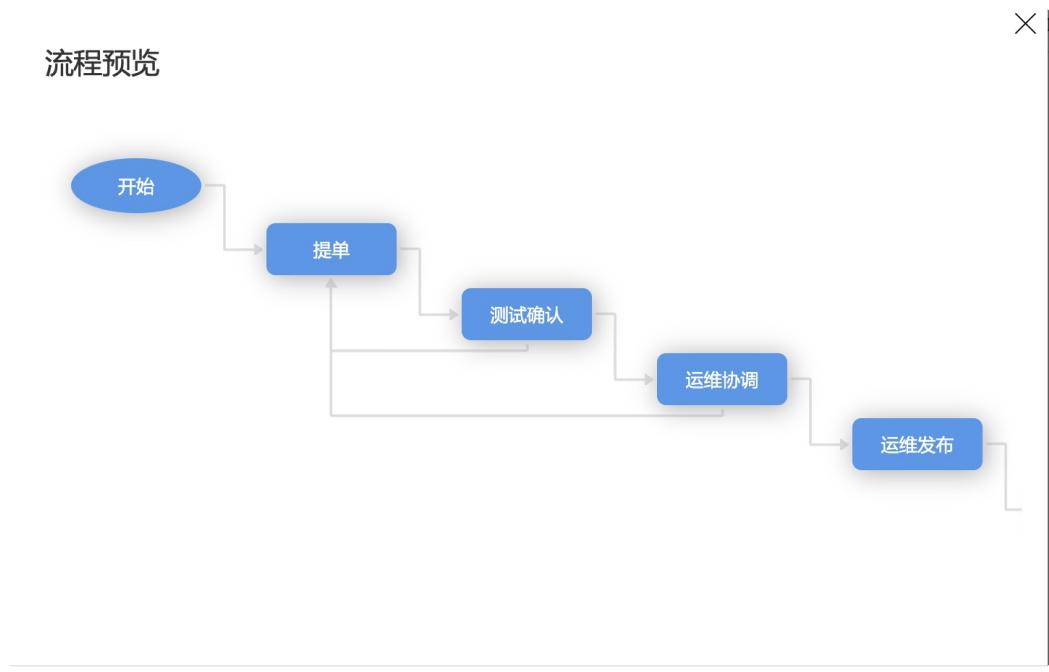
选择菜单【服务目录】，选中【根目录】，点击右侧【：】，点击【新增】，按提示新增一个名为应用发布的服务目录。

选中刚刚创建的服务目录【应用发布】，右侧会显示【添加】按钮，点击该按钮添加应用发布服务。



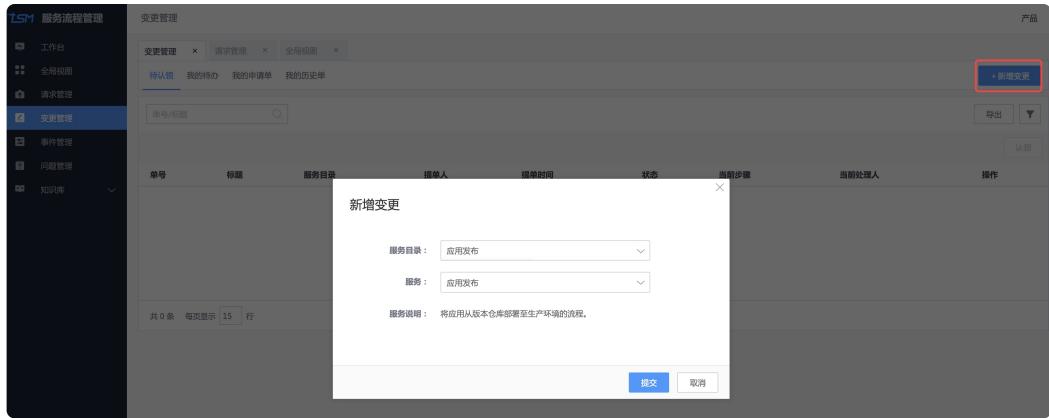
## 一次应用发布示例

创建完“应用发布”流程及服务目录后，接下来做一次应用发布演示。



### 产品提交发布需求

用 **产品** 账号登录 ITSM，选择【变更管理】菜单，点击【新增变更】，选择【应用发布】服务，点击【提交】。



在变更申请界面，填写本次版本发布的关键信息，包括版本包路径、MD5，用于自动化应用发布系统直接从仓库中获取版本，以及版本发布范围和时间。

点击【提交】，创建发布需求。



## 测试确认版本质量

测试人员收到一封待处理的测试确认邮件。

**【ITSM】变更管理单 【待处理】** ☆  
发件人: **breaking** <[blueking@63.com](mailto:blueking@63.com)>   
时 间: 2019年8月14日(星期三) 中午1:06  
收件人: **Pythoning** <[blueking@63.com](mailto:blueking@63.com)>

**【ITSM通知】变更管理**

Hi,您好!

您有1条变更管理单 【待处理】

标题: V5.1

单号: NO2019081413015540

服务目录: 应用发布

当前环节: 测试确认

当前处理人: ceshi(测试)

[点击查看详情](#)

如有任何问题, 可随时联系ITSM助手。

ITSM流程管理服务

2019年8月14日

使用测试账号登录 ITSM, 在待办列表中, 找到刚刚产品提的发布单。

The screenshot shows the ITSM Workbench interface. On the left, there's a sidebar with navigation links: 全局视图, 请求管理, 变更管理, 事件管理, 问题管理, and 知识库. The main area is titled '工作台' (Workbench). It has two tabs: '待认领列表' (Assigned Tasks) and '待办列表' (To-Do List), with '待办列表' being active. A red circle with the number '1' indicates a new task. Below the tabs is a table with the following data:

单号	标题	工单类型	申请人	提单时间	当前步骤
NO2019081413015540	V5.1	变更管理	blueking(blueking)	2019-08-14 13:01:55	测试确认

To the right of the table, there's a '最新动态' (Latest Activity) section with a message: '暂无动态 您目前还没有新动态更新!' (No new activity).

点击链接, 选择评估结果和测试结果, 并填写测试概况, 点击【通过】, 完成本环节流程。

The screenshot shows two main windows of the ITSM system:

- Ticket Detail Window:** A modal window titled "处理单 NO2019081413015540 标题: V5.1". It contains sections for "基本信息" (Basic Information) and "工单处理" (Work Order Processing). The "基本信息" section shows details like service type (变更管理), service name (应用发布), and submitter (blueking/blueking). The "工单处理" section includes a summary of the ticket's status and specific fields for "BUG修复" (Bug Fixing).
- Test Confirmation Dialog:** A modal titled "测试确认" (Test Confirmation) with tabs for "评估结果" (Evaluation Result) and "测试结果" (Test Result). It displays evaluation results (建议发布 / Recommended Release) and test results (通过 / Passed). Below these are detailed logs of test cases and their outcomes.

## 运维协调资源

使用运维的账号登录 ITSM，在待办列表找到单据。

The screenshot shows the ITSM Workbench interface with several key components:

- Left Sidebar:** A navigation tree with categories like Workbench, Request Management, Change Management, Event Management, Problem Management, Knowledge Base, Operation Data, Service Management, Service Catalog, Process Management, SLA Management, Role Configuration, and System Configuration.
- Pending List:** A table titled "待办列表" (Pending List) showing a single item: "NO2019081413015540" with title "V5.1", type "变更管理", submitter "blueking(blueking)", and creation time "2019-08-14 13:01:55". Status is listed as "运维协调" (Operational Coordination).
- Recent Activity:** A list of recent operations performed by "blueking" on the ticket, including processing steps and their timestamps.
- Operational Dashboards:** Three main dashboard panels:
  - 处理过的单据:** A bar chart showing the volume of tickets handled across categories: 请求管理 (7), 变更管理 (1), 事件管理 (0), 问题管理 (0), and 总计 (8).
  - 单据处理总耗时(h):** A donut chart showing the distribution of total processing time in hours: 0-1h (0.971h), 1-2h (0.423h), and 2-3h (0.006h).
  - 当前单据状态分布占比:** A pie chart showing the distribution of current ticket states: 待认领 (0%), 处理中 (25%), 挂起 (75%), and 待评价 (0%).

点击【通过】，开始协调发布资源，做好发布准备。

NO2019081413015540 标题：V5.1

操作人: NO2019081413015540 处理结果: 通过

版本类型: 全量 (发布至所有节点)

版本包路径: /etc/itsm/5k\_itsm\_2.1.14.tgz

发布范围: 一区-二区-三区-四区

停机发布: 是

计划发布时间: 2019-08-15 08:00:00

结束时间: 2019-08-15 10:00:00

版本日志: 新特性

新增欢乐模式  
礼物可以赠送给朋友  
**BUG修复**  
玩家阅读不对应先FPS帧数下降

处理时间: 2019-08-14 13:06:11

**测试确认** 处理人: ceshi(测试)

评估结果: 同意发布  
测试结果: 测试通过  
测试概况: 严重以上缺陷数: 0个  
测试用时: 8小时  
测试人员: ceshi, wj(A)

处理时间: 2019-08-14 13:12:10

**运维协调**

标准发布: •  是  否

**通过** 打印 保存字段

流转日志 通知关注人

- 2019-08-14 13:06:11 blueking(blueking) 处理节点【提单】(提交)
- 2019-08-14 13:12:10 ceshi(测试) 处理节点【测试确认】(通过)
- blueking(blueking) 处理节点【运维协调】(通过)

## 运维实施发布

完成了发布准备，到了计划发布时间，在自动化应用发布系统（如标准运维）上完成应用发布后，在该发布的运维发布环节填写本次运维发布的概况。

NO2019081413015540 标题：V5.1

操作人: NO2019081413015540 处理结果: 通过

测试确认 处理人: ceshi(测试)

评估结果: 同意发布  
测试结果: 测试通过  
测试概况: 严重以上缺陷数: 0个  
测试用时: 8小时  
测试人员: ceshi, wj(A)

处理时间: 2019-08-14 13:12:10

**运维协调** 处理人: blueking(blueking)

标准发布: 是  
处理时间: 2019-08-14 13:14:34

**运维发布**

引发故障:  是  否  
实际发布时间: •  2019-08-15 08:00:00 结束时间: •  2019-08-15 09:10:00  
运维操作时长: • 20分钟  
跟脚测试时长: • 50分钟  
发布结论: • 完成成功  
经验总结: • 传统应用发布模式仍然较慢，如果架构改造为微服务，使用K8S部署，可以更快，且实现不停机更新。  
发布单据: • 82837

**通过** 保存字段

流转日志 通知关注人

- 2019-08-14 13:06:11 blueking(blueking) 处理节点【提单】(提交)
- 2019-08-14 13:12:10 ceshi(测试) 处理节点【测试确认】(通过)
- 2019-08-14 13:14:34 blueking(blueking) 处理节点【运维协调】(通过)

## QC 发布评估

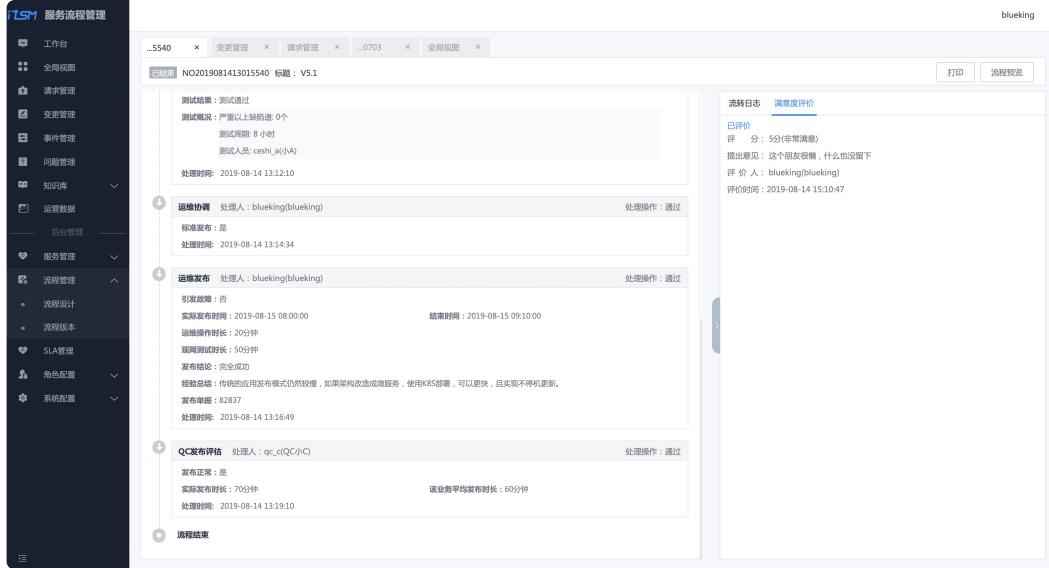
QC（质量保障）人员在工作台的待办列表中，找到本次发布的单据。

The screenshot shows the ITSM Service Management Platform's homepage. On the left is a dark sidebar with navigation links: 全局地图, 请求管理, 变更管理, 事件管理, 问题管理, and 知识库. The main area has a title '工作台' and a sub-section '待办列表'. A red box highlights a ticket entry: NO2019081413015540, 标题: V5.1, 工单类型: 变更管理, 提单人: blueking(blueking), 提单时间: 2019-08-14 13:01:55, 当前步骤: QC发布评估. To the right is a '最新动态' section with a message: '暂无动态 暂目前还没有新动态更新！' Below the dashboard are three circular charts: '处理过的单据' (1), '单据处理总耗时(h)' (empty), and '当前单据状态分布占比' (100%待认领, 0%待处理, 0%挂起, 0%待评价).

QC 根据业务指标监控系统的数据以及事件管理中是否存在关联的事件，对此次发布做出发布评估。

This screenshot shows the detailed view of ticket NO2019081413015540 for release V5.1. The ticket details include: 测试概况: 严重以上缺陷数: 0个, 测试周期: 8 小时, 测试人员: ceshi\_1(v/A), 处理时间: 2019-08-14 13:12:10. The '运维协调' section shows: 处理人: blueking(blueking), 处理操作: 通过, 标准发布: 是, 处理时间: 2019-08-14 13:14:34. The '运维发布' section shows: 处理人: blueking(blueking), 处理操作: 通过, 引发故障: 否, 实际发布时间: 2019-08-15 08:00:00, 结束时间: 2019-08-15 09:10:00, 运维操作时长: 20分钟, 跟踪测试时长: 50分钟, 失败缺陷: 完全成功, 经验总结: 传统的应用发布模式仍然较慢, 如果架构改造成微服务, 使用K8S部署, 可以更快, 目实现不停机更新, 发布单据: S2837, 处理时间: 2019-08-14 13:16:49. The 'QC发布评估' section at the bottom is highlighted with a red box, containing: 布局正常: • 是, 实际发布时长: • 70分钟, 该业务平均发布时长: 60分钟. Buttons for '通过' (Pass) and '保存字段' (Save Fields) are visible.

至此，一次应用发布的流程结束。



# 环境变更流程线上化

## 情景

变更是运维的职能之一，包含应用的配置、基础设施的环境等变更。

应用所依赖的服务器已经服役 5 年，不在维保期限，故障率抖升，为了保障应用的稳定性，运维团队计划在业务低高峰期，完成服务器的以旧换新。

ITSM 中的变更管理（Change Management）规范了变更环节的操作，确保高效有序的完成环境变更，对应用的可用性负责，并对生产环境的变更满足合规性检查。

变更管理是蓝鲸 ITSM 的一个模块，接下来介绍在蓝鲸 ITSM 如何完成这次服务器的替换升级。

## 前提条件

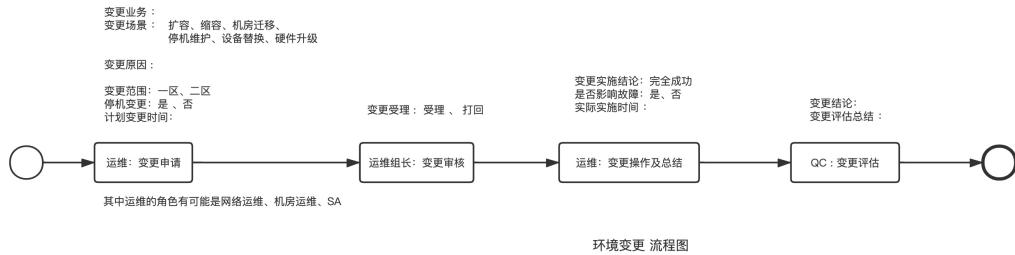
- 蓝鲸企业版，自带 ITSM SaaS。
- 准备环境变更流程中 **多个角色** 的账号，包含 **运维**、**QC**、**产品**，以及流程设计的 **管理员**。

## 操作步骤

- 梳理环境变更流程
- 创建环境变更服务目录及流程
- 一次环境变更示例

## 梳理环境变更流程

从 ITSM 理论出发，梳理环境变更流程图，包含运维变更申请、受理变更、变更操作和总结，以及最后质量保证（QC）对变更的评估管理。



## 创建环境变更服务目录及流程

先设计环境变更的流程，流程依附在服务目录上对外提供服务。

### 角色设置

参照 [角色设置](#) 完成对 [运维](#)、[产品](#) 和 [QC](#) 的授权。

### 设计环境变更流程

#### 填写流程信息

选择菜单【流程设计】，点击【新增】按钮，按提示填写流程信息。

ITSM 服务流程管理

新增设计流程

1 填写流程信息 2 定义与配置流程 3 流程启用设置

流程名称：\* 环境变更

流程说明：\* 应用的配置、基础设施的环境等变更  
至少输入5个字 16/100

流程类型：\* 变更

是否关联业务：\*  是  否

返回 下一步

流程类型选择【变更】，需要【关联业务】，因为环境变更和业务相关，同时关联业务对应的角色：产品、运维。

点击【下一步】，进入【定义与配置流程】环节。

### 定义与配置流程



点击上图【流程画布】中的【齿轮】，配置【提单】流程节点的字段。

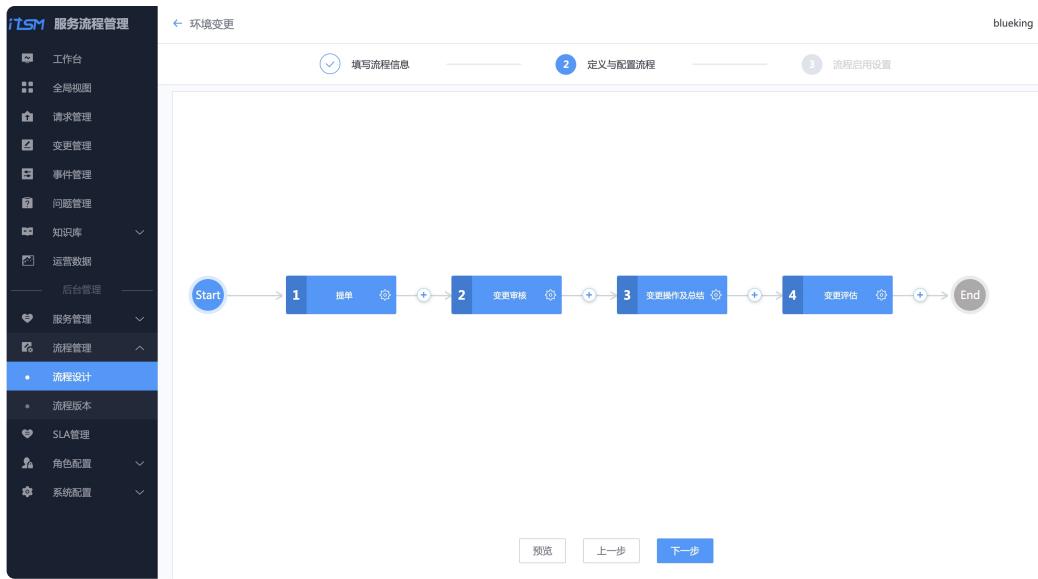
一般是运维提环境变更单据，所以操作角色选择【CMDB 业务公用角色】->【运维人员】(角色授权详见给角色分配权限)。

The screenshot shows the 'ITSM Service Flow Management' interface. On the left, there's a sidebar with various management options like Workstation, Global View, Request Management, Change Management, Event Management, Problem Management, Knowledge Base, Operation Data, and more. Under 'Process Management', 'Process Design' is selected. In the main area, the title is 'Configure Node'. The 'Basic Information' section has fields for 'Node Name' (填写) and 'Operational Role' (CMDB business public role). A red box highlights the 'Operational Role' dropdown. Below it is a table for adding fields, with one row shown: 'Field Name' (Title), 'Field Type' (Single-line text), 'Field Value' (Blank), 'Field Description' (Please enter the title), 'Field Validation Method' (Required), 'Layout Requirements' (Horizontal), 'Field Properties' (Custom), and 'Operations' (Move Up, Move Down, Edit). Buttons for 'Save' and 'Field Preview' are at the bottom.

点击【新增字段】，参照梳理环境变更流程，新增每个环节中需要的字段。

This screenshot shows the 'Edit Field' screen for the 'Change Scenario' field of the 'Ticket' node. The left sidebar is identical to the previous screenshot. The main area shows the 'Basic Information' section with 'Node Name' (Ticket) and 'Operational Role' (CMDB business public role). A red box highlights the '+ Add Field' button. To the right, the 'Field Display Name' (Change Scenario), 'Field Description' (This is the category of change), and 'Field Type' (Single-select dropdown) are set. Below these, under 'Data Source', 'Custom Data' is selected. A red box highlights the 'Custom Data' input field, which contains 'Expand', 'Shrink', and 'Other'. Other settings include 'Layout Requirements' (Horizontal), 'Field Validation Method' (Required), and 'Regular Expression' (None). Buttons for 'Save' and 'Cancel' are at the bottom.

参照梳理环境变更流程，完成整个环境变更流程的配置。



## 启用流程

【启用流程】，选择适合的通知策略，点击【提交】完成流程设计。

The screenshot shows the 'Process Activation Configuration' step. It includes sections for enabling the process and setting notification strategies. The 'ON' button for enabling the process is highlighted with a red box. The '邮件' (Email) and '微信' (WeChat) checkboxes under notification types are also highlighted with a red box.

## 流程模板实例化

选择菜单【流程设计】，找到刚编辑的环境变更流程，点击【部署】，生成流程实例。

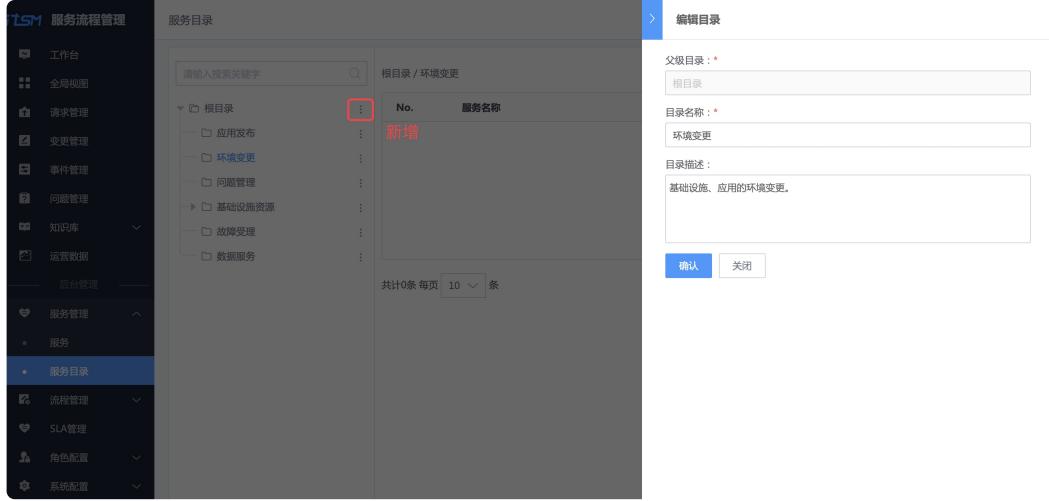
The screenshot shows the ITSM Service Flow Management interface. On the left, there's a sidebar with various management categories like Workstation, Global View, Request Management, Change Management, Event Management, Problem Management, Knowledge Base, Operation Data, and Backend Management. Under 'Flow Design', there are sub-options for Flow Version, SLA Management, Role Configuration, and System Configuration. The main area is titled 'Flow Design' and shows a table of existing flows. One row for 'Environment Change' is selected, and its operation menu (Edit, Export, Deploy, Delete) is visible. A modal dialog box is overlaid on the page, prompting the user to confirm the deployment of the selected flow.

## 在服务目录中新增"环境变更"服务，并绑定流程

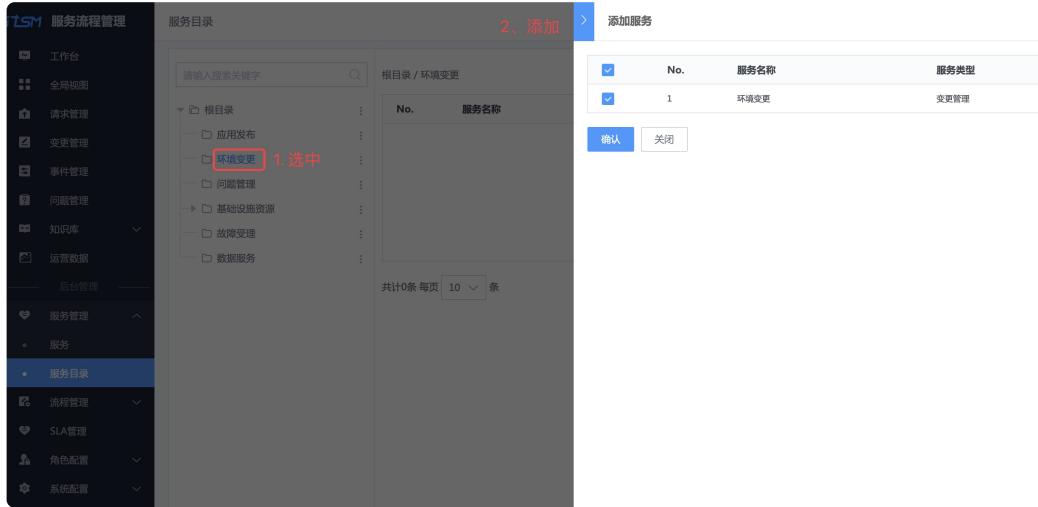
选择菜单【服务】，点击【新增】按钮，新增“环境变更”服务，并关联刚生成的流程实例。

The screenshot shows the ITSM Service Management interface. On the left, there's a sidebar with various management categories like Workstation, Global View, Request Management, Change Management, Event Management, Problem Management, Knowledge Base, Operation Data, and Backend Management. Under 'Service', there are sub-options for Service Catalog, Flow Management, SLA Management, Role Configuration, and System Configuration. The main area is titled 'Service' and shows a table of existing services. A modal dialog box is overlaid on the page, titled 'Add Service', where a new service named 'Environment Change' is being created. The dialog includes fields for 'Associate Flow Version' (set to 'Environment Change (20190815192807)'), 'Service Type' (set to 'Change Management'), 'Service Level' (set to '四级 - Level 4'), and a 'Service Description' field containing the note 'The configuration, infrastructure environment, etc.'.

选择菜单【服务目录】，选中【根目录】，点击右侧【：】，点击【新增】，按提示新增一个名为环境变更的服务目录。



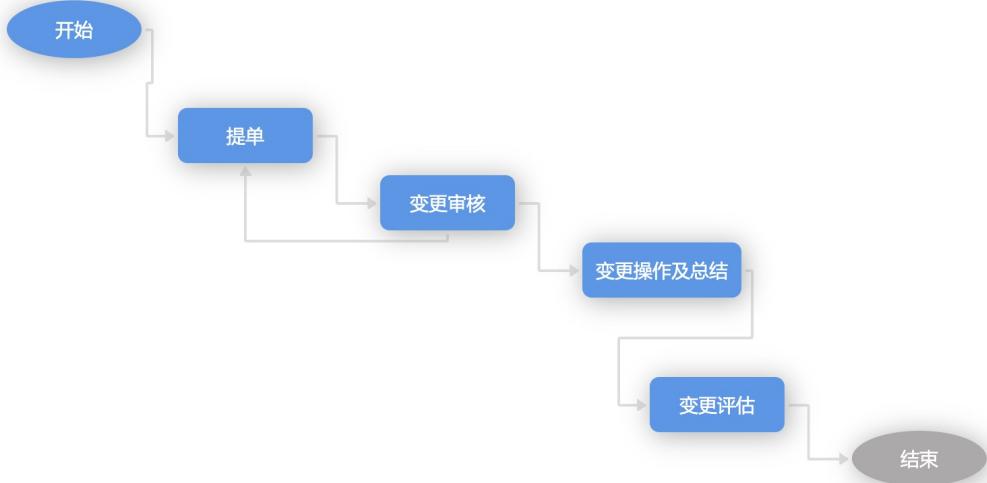
选中刚刚创建的服务目录【环境变更】，右侧会显示【添加】按钮，点击该按钮添加环境变更服务。



至此，流程设计和服务目录已新建好，接下来做一次环境变更演示。

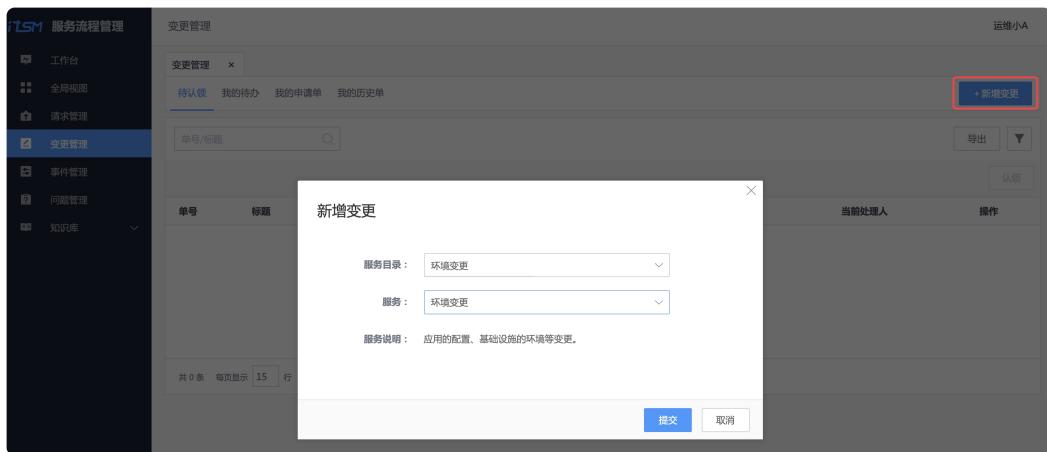
## 一次环境变更示例

## 流程预览



### 运维：变更申请

用[运维](#)账号登录ITSM，选择【变更管理】菜单，点击【新增变更】，选择【环境变更】服务，点击【提交】。



在【新增变更】界面，填写本次环境变更的关键信息，包括变更原因、范围以及时间等。

点击【提交】，完成变更申请。

ITSM 服务流程管理

运维小A

变更申请

是否使用模板：  使用模板可以快速填写字段信息

标题：**一区、二区服务器以旧换新** 关联业务：**欢乐游戏(demo)**

变更场景：**硬件升级**

变更原因：**应用所依赖的服务器已经服役5年，不在维保期限，故障率飙升，为了保障应用的稳定性，运维团队计划在业务低峰期，完成服务器的以旧换新。**

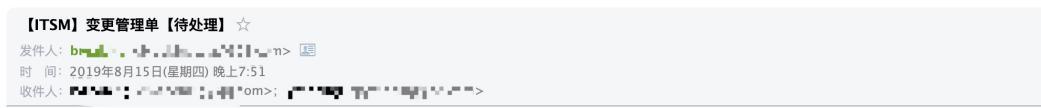
变更范围：**一区,二区** 停机变更：**是**

计划变更时间：**2019-08-15 20:00:00** 结束时间：**2019-08-15 21:00:00**

**提交** **保存模板** **取消**

## 运维组长：变更审核

运维组长收到一封待处理的变更审核邮件。



### 【ITSM通知】变更管理

HI,您好!

您有1条变更管理单【待处理】

标题: **一区、二区服务器以旧换新**

单号: NO2019081519494023

服务目录: 环境变更

当前环节: 变更审核

当前处理人: blueking(blueking),ops\_a(运维小A)

[点击查看详情](#)

如有任何问题, 可随时联系ITSM助手。

ITSM流程管理服务

2019年8月15日

使用运维账号登录 ITSM，在待办列表中，找到待处理的环境变更单。

ITSM 服务流程管理

工作台

待办列表 1

单号	标题	工单类型	提交人	提交时间	当前步骤
NO2019081519494023	一区、二区服务器以旧换新	变更管理	ops_a(运维小A)	2019-08-15 19:49:40	变更审核

最新动态

- 14秒前 NO2019081519494023 ops\_a(运维小A) 处理节点【提单】(提交)
- 5分钟前 NO2019081519403742 ops\_a(运维小A) 处理节点【变更操作及总结】(通过)
- 6分钟前 NO2019081519403742 ops\_a(运维小A) 处理节点【变更审核】(通过)
- 7分钟前 NO2019081519403742 ops\_a(运维小A) 处理节点【提单】(提交)
- 2小时前 NO2019081517062115 ops\_a(运维小A) 处理节点【故障总结】(通过)
- 2小时前 NO2019081517062115 ops\_a(运维小A) 处理节点【故障跟进】(通过)

处理过的单据

单据处理总耗时(h)

当前单据状态分布占比

点击链接，审核本次变更，点击【通过】，完成本环节流程。

ITSM 服务流程管理

工作台

全局视图 x 变更管理 x 请求管理 x 全局视图 x ...3742 x

运维小A

处理中 NO2019081519494023 标题：一区、二区服务器以旧换新 挂起 撤单 打印 流程预览

**基本信息**

工单类型： 变更管理 提单人： ops\_a(运维小A) / 18111112222 服务目录： 环境变更  
提单人部门： ..

当前处理人： blueking(blueking),ops\_a(运维小A)  
提单时间： 2019-08-15 19:49:40

**工单处理**

**提单** 处理人：ops\_a(运维小A) 关联业务：欢乐游戏(demo)  
标题：一区、二区服务器以旧换新 变更场景：硬件升级  
变更原因：应用所依赖的服务器已经服役5年，不在维保期限，故障率飙升，为了保障应用的稳定性，运维团队计划在业务低峰期，完成服务器的以旧换新。  
变更范围：一区-二区  
计划变更时间：2019-08-15 20:00:00  
结束时间：2019-08-15 21:00:00  
处理时间：2019-08-15 19:51:41

**变更审核**

变更受理： 受理 通过 打回 保存字段

## 运维：变更操作及总结

环境变更完毕后，填写变更总结。

LSM 服务流程管理

..4023 x

处理单 NO2019081519494023 标题：一区、二区服务器以旧换新 远维小A

当前处理人：blueking(blueking).ops\_a(远维小A)

提单时间：2019-08-15 19:49:40

**工单处理**

**提单** 处理人：ops\_a(远维小A) 处理操作：提单

标题：一区、二区服务器以旧换新  
关联业务：欢乐游戏(demo)  
变更场景：硬件升级  
变更原因：应用所依赖的服务器已经服役5年，不在维保期限，故障率较高，为了保障应用的稳定性，运维团队计划在业务低高峰期，完成服务器的以旧换新。  
变更范围：一区、二区  
计划变更时间：2019-08-15 20:00:00  
停机变更：是  
结束时间：2019-08-15 21:00:00  
处理时间：2019-08-15 19:51:41

**变更审核** 处理人：ops\_a(远维小A) 处理操作：通过

变更类型：受控  
处理时间：2019-08-15 19:53:06

**变更操作及总结**

变更实施结论：完全成功  
是否引发故障：是  
计划变更时间：2019-08-15 20:00:00  
结束时间：2019-08-15 20:30:00

通过 保存字段

## QC：变更评估

QC（质量保障）人员在工作台的待办列表中，找到本次环境变更的单据。

LSM 服务流程管理

工作台 远维小A

待认领列表 待办列表 1

单号	标题	工单类型	提单人	提单时间	当前步骤
NO2019081519494023	一区、二区服务器以旧换新	变更管理	ops_a(远维小A)	2019-08-15 19:49:40	变更评估

**最新动态**

- 4秒前 NO2019081519403742 qc\_z(QC小D) 处理节点【变更评估】(通过)
- 2小时前 NO2019081517062115 qc\_z(QC小D) 处理节点【故障评估】(通过)
- 1天前 NO2019081413015540 qc\_z(QC小D) 处理节点【QC发布评估】(通过)

QC 依据该业务的业务关键指标数据、投诉情况，做出变更评估。

The screenshot displays the ITSM Service Flow Management system. A specific change request (ID: 4023) is being viewed. The request title is 'NO2019081519494023 标题：一区、二区服务器以旧换新'. The workflow steps shown are:

- 变更审核**: 处理人: ops\_a(运维小A), 处理操作: 通过. 变更实施结论: 完全成功. 实际变更时间: 2019-08-15 20:00:00. 结束时间: 2019-08-15 21:00:00.
- 变更操作及总结**: 处理人: ops\_a(运维小A), 处理操作: 通过. 变更实施结论: 完全成功. 实际变更时间: 2019-08-15 20:00:00. 结束时间: 2019-08-15 20:30:00.
- 变更评估**: 变更结论: 完全成功. 变更评估总结: 符合变更预期. 处理操作: 通过. 处理时间: 2019-08-15 19:53:51.

On the right side, there is a '流转日志' (Flow Log) section showing the following activities:

- 2019-08-15 19:51:41: ops\_a(运维小A) 处理节点【提单】(提交)
- 2019-08-15 19:53:06: ops\_a(运维小A) 处理节点【变更审核】(通过)
- 2019-08-15 19:53:51: ops\_a(运维小A) 处理节点【变更操作及总结】(通过)

至此，一次环境变更的流程结束。

This screenshot shows a fault report (工单) for '欢乐游戏(demo)'. The basic information includes:

- 工单类型: 变更管理
- 申请人: ops\_a(运维小A) / 18111112222
- 当前处理人: ...
- 下单时间: 2019-08-15 19:49:40

The workflow steps shown are:

- 提单**: 处理人: ops\_a(运维小A). 处理操作: 提单. 处理业务: 欢乐游戏(demo).
- 变更审核**: 处理人: ops\_a(运维小A), 处理操作: 通过. 变更实施结论: 完全成功. 实际变更时间: 2019-08-15 20:00:00. 结束时间: 2019-08-15 21:00:00.
- 变更操作及总结**: 处理人: ops\_a(运维小A), 处理操作: 通过. 变更实施结论: 完全成功. 实际变更时间: 2019-08-15 20:00:00. 结束时间: 2019-08-15 20:30:00.
- 变更评估**: 处理人: qc\_c(QC小C), 处理操作: 通过. 变更结论: 完全成功. 变更评估总结: 符合变更预期. 处理时间: 2019-08-15 19:55:30.

## 故障提报流程线上化

### 情景

ITSM 中的故障管理(Incident Management)帮企业构建了一套应对故障的处理机制，确保故障来临之时可以高效有序的响应、处理以及回溯故障的表面和本质，并将根源问题通过转单到问题管理模块，实现故障的闭环。

故障管理是蓝鲸 ITSM 内置的模块之一，接下来介绍在蓝鲸 ITSM 是如何应对部分地区用户集中无法访问网站的投诉。

## 前提条件

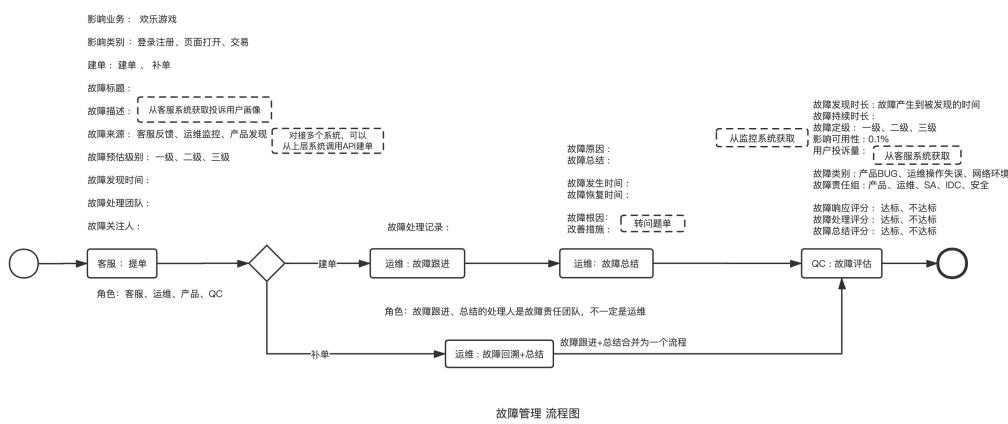
- 蓝鲸企业版，自带 ITSM SaaS。
- 准备故障汇报流程中 **多个角色** 的账号，包含 **运维**、**QC**、**产品**，以及流程设计的 **管理员**。

## 操作步骤

- 梳理故障汇报流程
- 创建故障汇报服务目录及流程
- 一次故障汇报示例

### 梳理故障汇报流程

从 ITSM 理论出发，梳理故障汇报的流程图，包含客服 提单、运维 故障跟进、运维 回溯总结故障，以及最后质量保证(QC)对故障的 评估管理。



“  
流程图中是一个实践案例，部分数据需要从周边系统获取，此处功能需要做二次开发，本教程专注流程本身。  
”

### 创建故障汇报服务目录及流程

先设计故障汇报的流程，流程依附在服务目录上对外提供服务。

## 角色设置

参照 角色设置 完成对 运维、产品 和 QC 的授权。

## 设计故障受理流程

### 填写流程信息

选择菜单【流程设计】，点击【新增】按钮，按提示填写流程信息。



ITSM 服务流程管理

新增设计流程

1 填写流程信息 2 定义与配置流程 3 流程启用设置

blueking

流程名称：\* 故障受理

流程说明：\* 应对生产环境故障的流程。  
至少输入5个字 12/100

流程类型：\* 事件

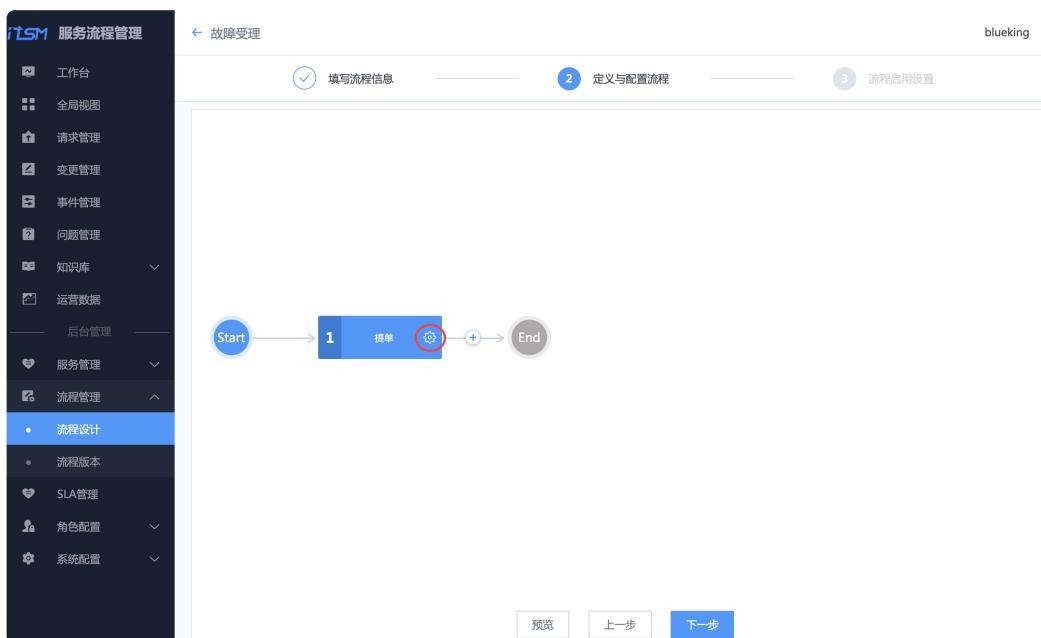
是否关联业务：\*  是  否

返回 下一步

流程类型选择【事件】，需要【关联业务】，因为故障提报和业务相关，同时关联业务对应的角色：产品、运维。

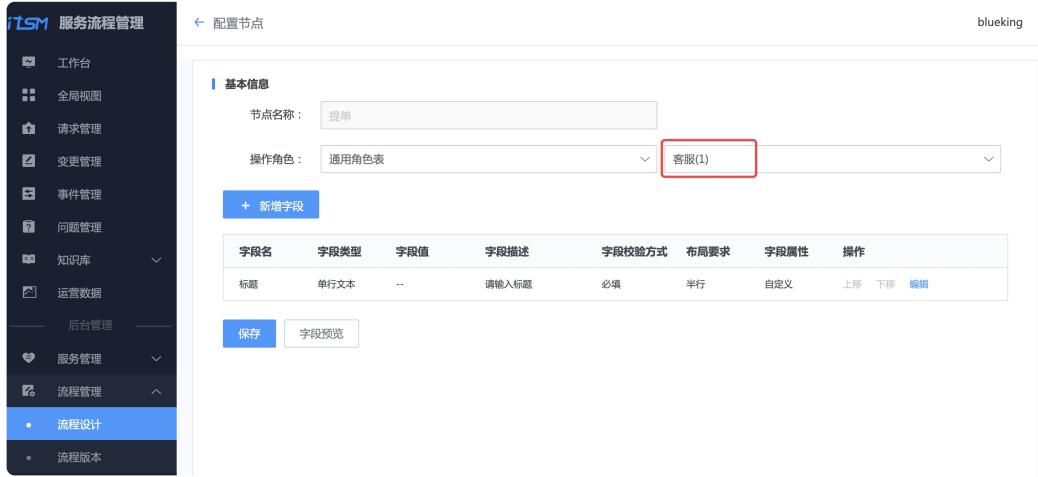
点击【下一步】，进入【定义与配置流程】环节。

### 定义与配置流程

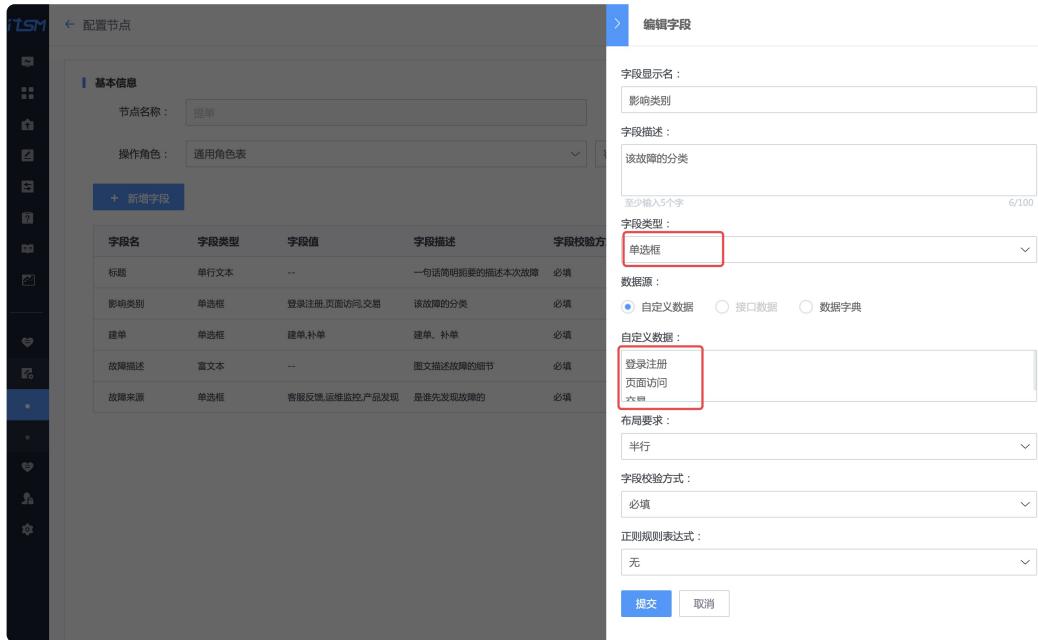


点击上图【流程画布】中的【齿轮】，配置【提单】流程节点的字段。

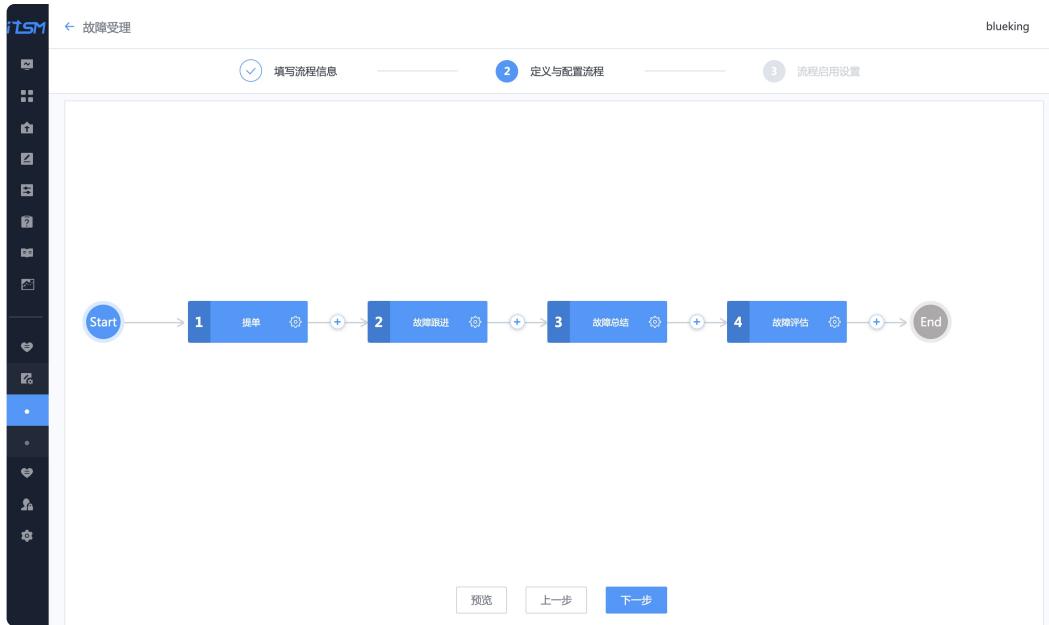
一般是客服提故障单据，所以操作角色选择【通用角色表】->【客服】(客服角色授权详见给角色分配权限)。



点击【新增字段】，参照梳理故障提报流程，新增每个环节中需要的字段。



参照梳理故障提报流程，完成整个故障受理流程的配置。



## 启用流程

【启用流程】，选择适合的通知策略，点击【提交】完成流程设计。

**设置流程启用管理**

是否启用该流程： ON

**设置任务通知策略**

是否通知： 是  否

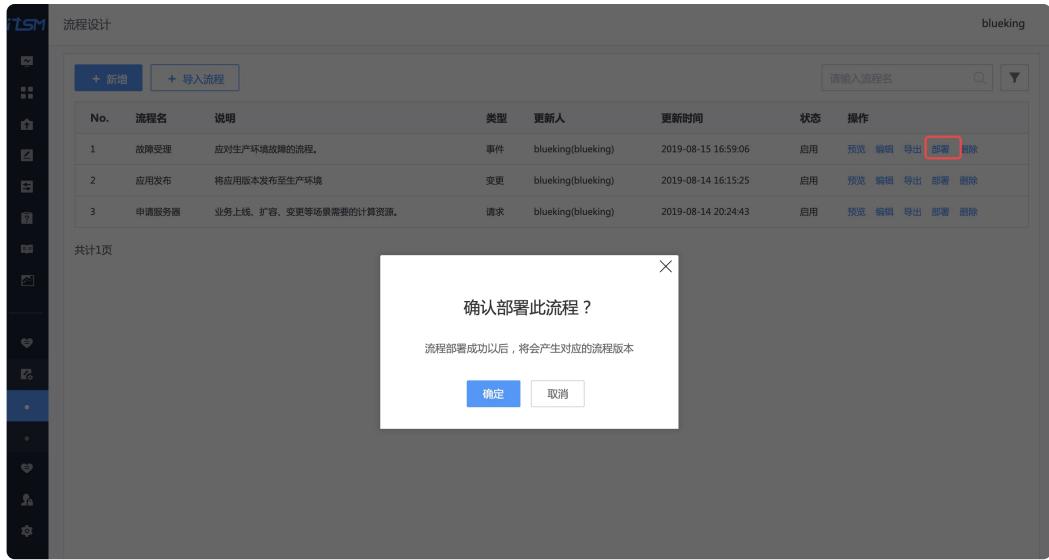
是否督办： 是  否

通知频率： 首次通知，以后不再通知  
 首次通知后，次日起每天定时通知

通知类型： SMS短信  邮件  微信

## 流程模板实例化

选择菜单【流程设计】，找到刚编辑的故障受理流程，点击【部署】，生成流程实例。

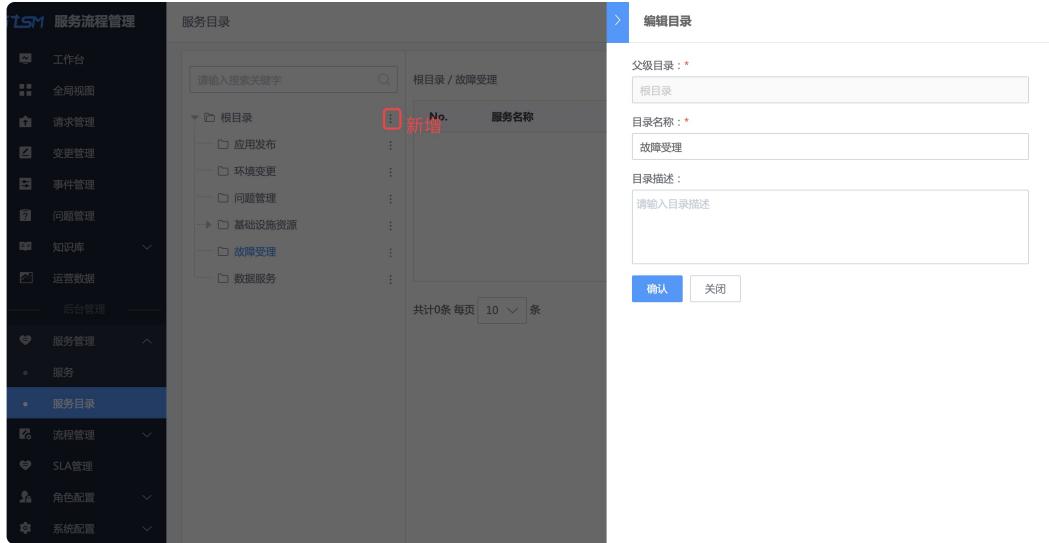


### 在服务目录中新增"故障受理"服务，并绑定流程

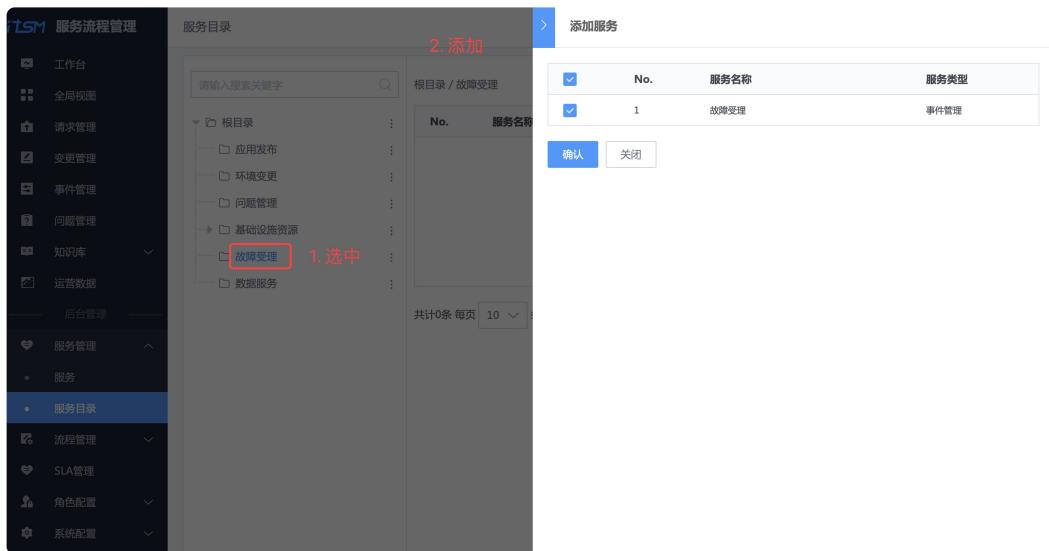
选择菜单【服务】，点击【新增】按钮，新增"故障受理"服务，并关联刚生成的流程实例。



选择菜单【服务目录】，选中【根目录】，点击右侧【：】，点击【新增】，按提示新增一个名为故障受理的服务目录。



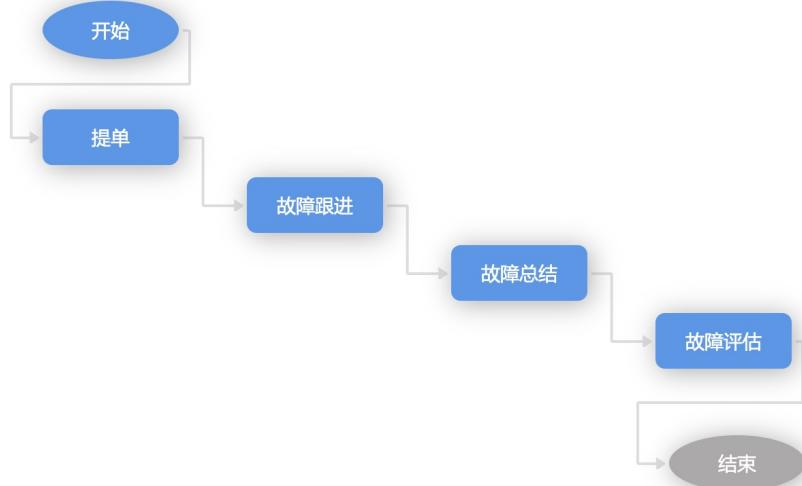
选中刚刚创建的服务目录【故障受理】，右侧会显示【添加】按钮，点击该按钮添加故障受理服务。



至此，流程设计和服务目录已新建好，接下来做一次故障汇报演示。

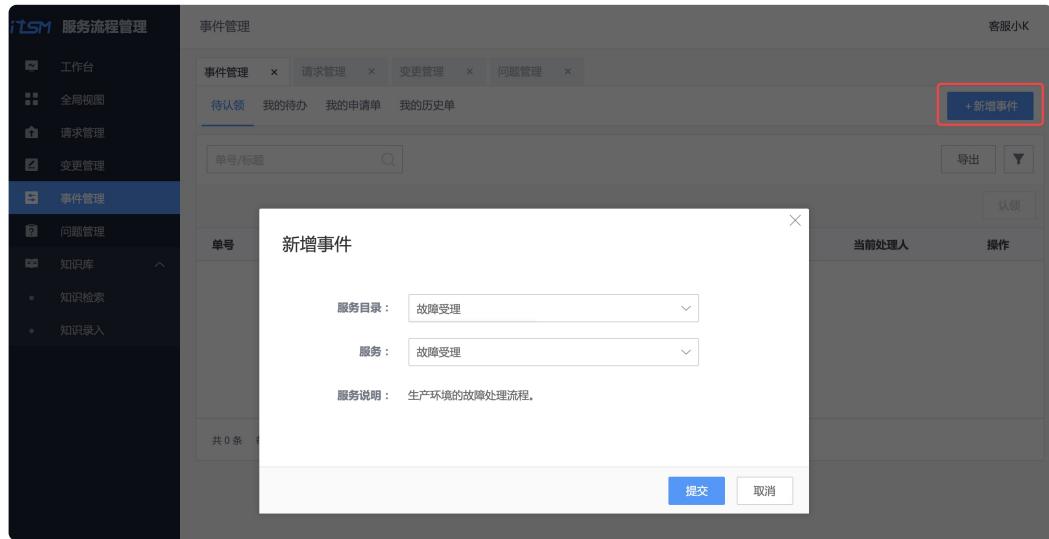
## 一次故障汇报示例

## 流程预览



### 客服提交故障单

用 **客服** 账号登录 ITSM，选择【事件管理】菜单，点击【新增事件】，选择【故障受理】服务，点击【提交】。



在【新增事件】界面，填写本次故障报的关键信息，包括故障标题、截图以及故障处理团队等。

点击【提交】，报故障。

ITSM 服务流程管理

客服小K

### 事件申请

是否使用模板：  使用模板可以快速填写字段信息

标题：\* 北方地区的某运营商用户，集中无法访问A网站 关联业务：\* 欢乐游戏(demo)

影响类别：\*  登录注册  页面访问  交易 建单：\*  建单  补单

故障描述：\*

从今天17:00开始，陆续收到北方地区的某运营商用户，集中无法访问网站。  
截图如下，疑似该运营商的DNS未解析A网站的域名

**无法访问此网站**

拒绝了我们的连接请求。  
请试试以下办法：  
 检查网络连接

Markdown WYSIWYG

故障来源：\*  客服反馈  运维监控  产品发现 故障预估级别：\*  一级  二级  三级

故障发现时间：\*  2019-08-15 17:00:00 故障处理团队：\* ops\_a(运维小A)

故障关注人：\* chanpin(产品)

**提交** **保存模板** **取消**

## 运维跟进故障处理

运维收到一封待处理的故障受理邮件。

**【ITSM】事件管理单【待处理】** ☆

发件人: **breaking** <[REDACTED]@tom>

时间: 2019年8月15日(星期四)下午5:20

收件人: [REDACTED]

**【ITSM通知】事件管理**

Hi,您好!

您有1条事件管理单【待处理】

标题: 北方地区的某运营商用户, 集中无法访问A网站

单号: NO2019081517062115

服务目录: 故障受理

发现时间: --

当前环节: 故障跟进

当前处理人: blueking(blueking),ops\_a(运维小A)

[点击查看详情](#)

如有任何问题, 可随时联系ITSM助手。

ITSM流程管理服务

2019年8月15日

使用运维账号登录 ITSM, 在待办列表中, 找到待处理的故障单。

ITSM 服务流程管理

工作台

全局视图  
请求管理  
变更管理  
**事件管理**  
问题管理  
知识库

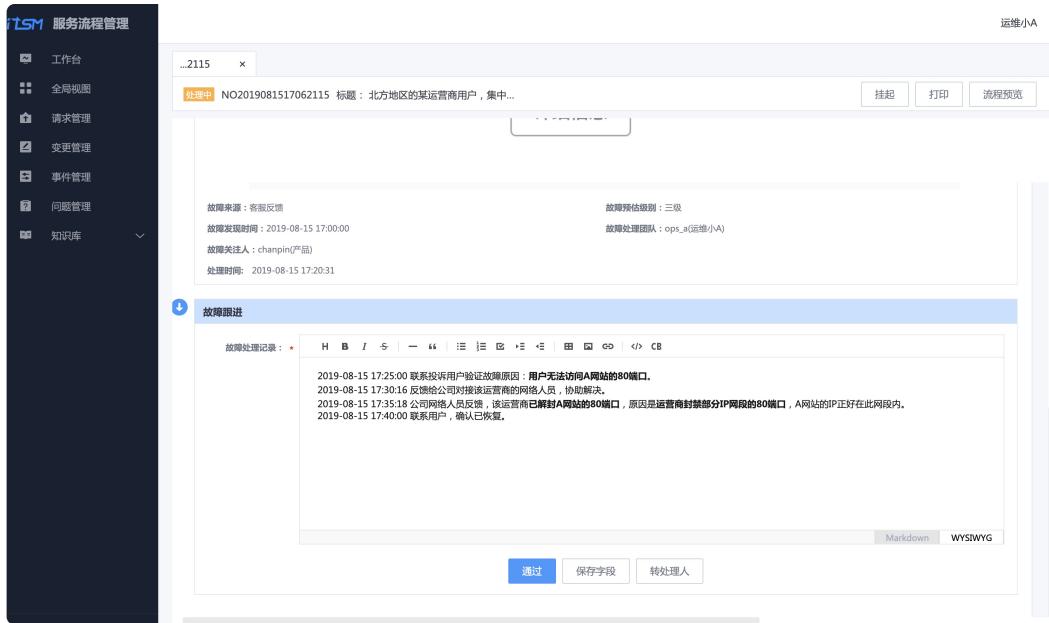
待认领列表 **待办列表** ①

单号	标题	工单类型	提单人	提单时间	当前步骤
NO2019081517062115	北方地区的某运营商用户, 集中无法访问A网站	事件管理	kefu(客服小K)	2019-08-15 17:06:21	故障跟进

最新动态  
暂无动态  
您目前还没有新动态更新 !

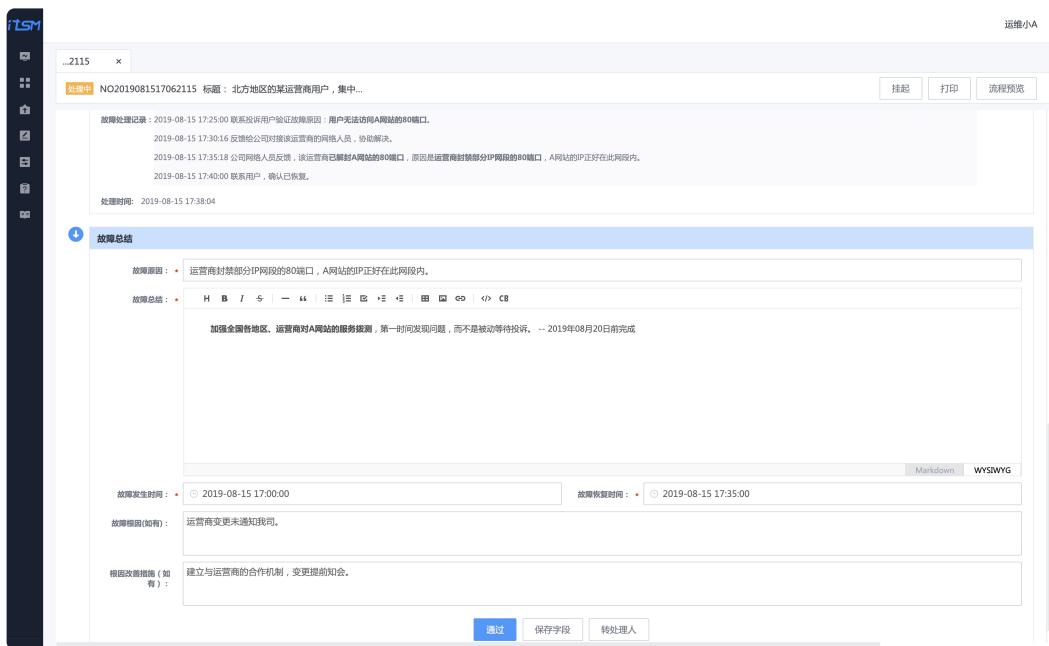
运维小A

点击链接, 填写故障处理记录, 点击【通过】, 完成本环节流程。



## 故障总结

处理完故障后，运维复盘本次故障，做故障总结。



## QC 做故障评估

QC（质量保障）人员在工作台的待办列表中，找到本次故障的单据。

ITSM 工作台

待认领列表 | 待办列表 1

单号	标题	工单类型	提单人	提单时间	当前步骤
NO2019081517062115	北方地区的某运营商用户，集中无法访...	事件管理	kefu(客服小K)	2019-08-15 17:06:21	故障评估

最新动态  
1天前 NO2019081413015540  
qc\_c(QC小C) 处理节点【QC发布评估】(通过)

QC 依据本次故障的影响范围、原因以及处理情况，做出故障评估。

ITSM

...2115 x

处理中 NO2019081517062115 标题：北方地区的某运营商用户，集中...

处理时间：2019-08-15 17:38:04

**故障总结** 处理人：ops\_a(运维小A) 处理操作：通过

故障原因：运营商对禁部分IP网段的80端口，A网站的IP正好在此网段内。  
故障总结：加强全国各地区、运营商对A网站的服务监测，第一时间发现问题，而不是被动等待投诉。-- 2019年08月20日前完成

故障发生时间：2019-08-15 17:00:00 故障恢复时间：2019-08-15 17:35:00

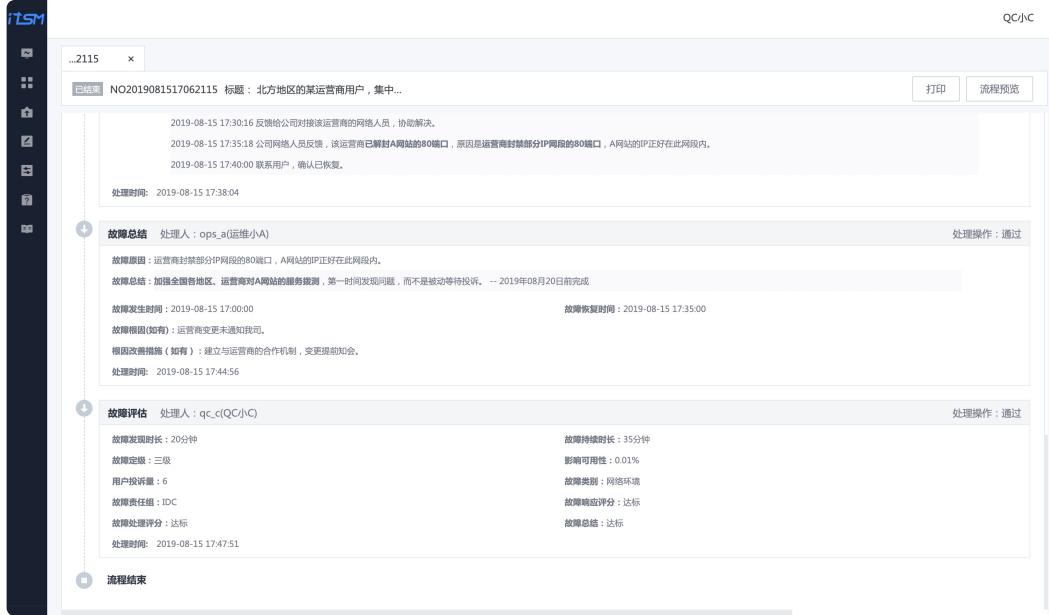
故障根因(如何)：运营商变更未通知我司。  
根因改善措施(如有)：建立与运营商的合作机制，变更提前告知。  
处理时间：2019-08-15 17:44:56

**故障评估**

故障发现时间： 故障持续时间：  
故障定级： 一级  二级  三级  
影响可用性：  
用户投诉量： 故障类别：  
故障责任组： 故障响应评分： 达标  不达标  
故障处理评分： 达标  不达标  
故障总结：

通过 | 保存字段 | 转处理人

至此，一次故障提报和处理的流程结束。



## 常见问题

### 蓝鲸支持私有化部署吗

蓝鲸支持私有化部署。

私有化部署，满足企业对敏感数据的合规性要求以及自主掌控的能力。

### 蓝鲸的代码开放吗

目前蓝鲸开源了 PaaS、CMDB、标准运维、BCS、蓝鲸 CI，更多开源计划请关注蓝鲸公众号。



## 蓝鲸的 PaaS 是真 PaaS 吗

蓝鲸的 **PaaS 平台** 是一个开放的平台，让用户可以简单、快速地创建、部署和管理应用，它提供了完善的前后台开发框架、服务总线（ESB）、调度引擎、公共组件等模块，帮助用户快速、低成本、免运维地构建支撑工具和运营系统。

PaaS 平台为一个应用从创建到部署，再到后续的维护管理提供了完善的自助化和自动化服务，如日志查询、监控告警等，从而使用户可以将全部精力投入到应用的开发之中。

PaaS 平台的主要功能有：支持多语言的开发框架/样例、免运维托管、SaaS 运营数据可视化、企业服务总线（API Gateway）、可拖拽的前端服务（MagicBox）等。