

# 第三次作业-基于 JOB 作业平台实现 游戏主机日志备份

## 一、 实验目的

- (1) 了解蓝鲸作业平台的功能与使用方法
- (2) 掌握作业平台的基本概念和作业平台的基本使用
- (3) 熟悉脚本的概念和语法，学习简单脚本的编写
- (4) 掌握作业平台 “方案执行”、“获取作业执行状态” 等接口的调用
- (5) 熟悉 Django 模型的设计和 ORM 操作，完成备份记录的建模和数据库读写功能
- (6) 提升 SaaS 应用的开发技能，巩固 Django 基础知识

## 二、 实验环境

- (1) 硬件环境需求： PC 或笔记本， 支持外网访问
- (2) 软件环境需求

系统： Windows, MacOS, Linux

安装 Python 3.6.12

安装 MySQL 8.3

安装 Git (最新版本即可)

安装 pre-commit 代码检查工具 (可选)

安装 VSCode, PyCharm 或其它 IDE

### 三、 实验内容

在《基于 CMDB 配置平台管理游戏的主机》的基础上，借助蓝鲸作业平台编写脚本，通过蓝鲸网关/ESB 组件 API 调用作业平台接口实现文件的查询和备份，并根据文件查询和备份条件，设计对应接口；同时记下备份记录，实现备份记录的查询功能。

### 四、 实验评分标准

**整体要求：**请同学们采用迭代方式进行需求分析、面向对象设计和编程实现，实训课报告中需包含相应的需求规约、设计规约、接口文档，项目开发说明

**考点一：**在作业平台编写简单脚本，实现文件查询和文件备份执行方案



步骤类型: 执行脚本  
步骤名称: list\_file  
脚本来源: 手工录入  
脚本内容:

```
Python
/
8 def list_file(path, indent=0):
9     names = []
10    cnt = 0
11    total_size = 0
12    if not os.path.exists(path):
13        data = {"message": "root_path not exists."}
14        return data
15    for filename in os.listdir(path):
16        full_path = os.path.join(path, filename)
17        suffix = "." + sys.argv[2]
18        if os.path.isfile(full_path) and filename.endswith(suffix):
19            names.append(filename)
20            cnt += 1
21            size = os.path.getsize(full_path)
22            total_size += size
23    names = ";".join(names)
24    data = {"bk_file_list": names, "bk_file_cnt": cnt, "bk_file_total_size": total_size}
25    return data
26
27 if __name__ == "__main__":
28     root_path = sys.argv[1]
29     data = list_file(root_path)
30     print(json.dumps(data, indent=4))
```

脚本参数: \${root\_path} \${suffix}  
超时时长: 7200 (s)  
错误处理: 不忽略  
执行账号: root  
执行目标: host\_list

相关资料:

1.蓝鲸作业平台: [蓝鲸作业平台](#)

**考点二:** 在《基于 CMDB 配置平台管理游戏的主机》的基础上, 实现文件查询接口, 通过蓝鲸 ESB 组件 API 联通作业平台进行方案执行和结果查询, 并在前端进行参数传递和数据渲染



相关资料：

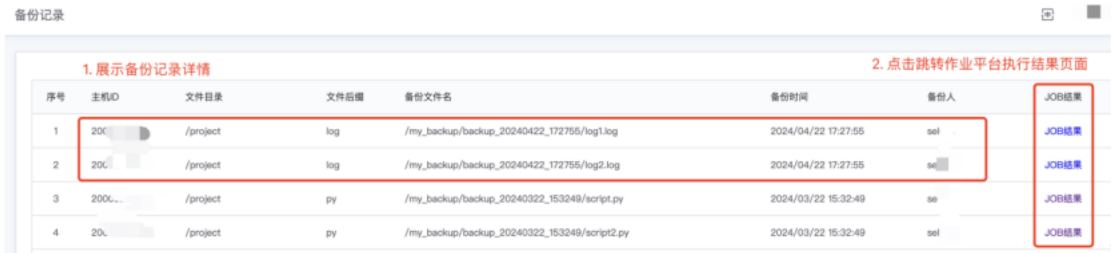
1.蓝鲸组件 API 文档：[蓝鲸组件 API 文档中心](#)

2.蓝鲸 MagicBox 组件库：[蓝鲸 MagicBox 组件库](#)

**考点三：**实现文件备份接口，通过蓝鲸 ESB 组件 API 联通作业平台进行方案执行和结果查询，记下备份记录，并在前端进行参数传递和数据渲染



**考点四：**实现备份记录查询接口，展示备份主机、备份文件目录、备份文件后缀等信息，并附上备份执行结果链接，点击可跳转作业平台对应的执行方案结果页面



其他评分项:

- 1. Python 代码符合 [PEP8 规范](#)，可酌情加分
- 2. 系统边界考虑完善，系统性能优良，可酌情加分
- 3. 前端界面优美，用户交互体验良好，可酌情加分
- 4. 实现主机状态拉取、主机批量选择等功能，可酌情加分
- 5. 使用异步任务进行文件备份，可酌情加分
- 6. 后端代码能够实现单元测试以及日志、异常处理等，可酌情加分

五、 实验过程与结果

1. 创建作业

(1) 创建文件查询作业

作业平台

作业管理

作业列表

作业详情

查看作业详情: [1111141012] 作业详情

作业名称: [1111141012] 文件查询

作业描述: --

作业语言: Python

作业路径: /usr/bin/python3

作业参数: --

作业环境: host\_bin

作业资源: 1GB 内存 / 1核 CPU

作业状态: 运行中

作业日志: 查看作业日志

查看作业步骤

作业名称: 文件查询

作业ID: 1111141012

作业语言: Python

作业路径: /usr/bin/python3

作业参数: --

作业环境: host\_bin

作业资源: 1GB 内存 / 1核 CPU

作业状态: 运行中

作业日志: 查看作业日志

(2) 创建文件备份作业

作业平台

作业管理

作业列表

作业详情

查看作业详情: [1111141012] 作业详情

作业名称: [1111141012] 文件备份

作业描述: --

作业语言: Python

作业路径: /usr/bin/python3

作业参数: --

作业环境: host\_bin

作业资源: 1GB 内存 / 1核 CPU

作业状态: 运行中

作业日志: 查看作业日志

查看作业步骤

作业名称: 文件备份

作业ID: 1111141012

作业语言: Python

作业路径: /usr/bin/python3

作业参数: --

作业环境: host\_bin

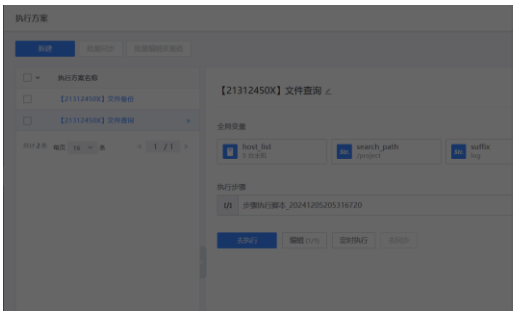
作业资源: 1GB 内存 / 1核 CPU

作业状态: 运行中

作业日志: 查看作业日志

2. 创建执行方案

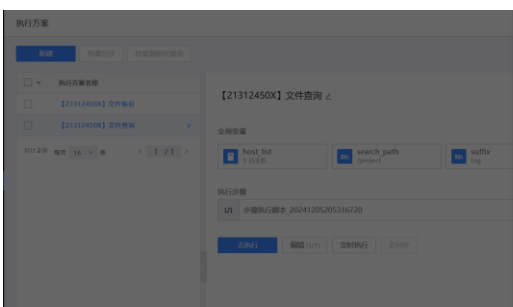
(1) 创建文件查询执行方案



|       |           |
|-------|-----------|
| 方案类型  | 主机列表      |
| 方案名称  | host_list |
| 方案组   | 5 台主机     |
| 方案描述  |           |
| 执行时必填 | 是         |

| IPv4       | IPv6 | Agent 状态 | OS 名称        |  |
|------------|------|----------|--------------|--|
| 10.0.48.45 | ---  | 正常       | linux centos |  |
| 10.0.48.34 | ---  | 正常       | linux centos |  |
| 10.0.48.48 | ---  | 正常       | linux centos |  |
| 10.0.48.37 | ---  | 正常       | linux centos |  |
| 10.0.48.32 | ---  | 正常       | linux centos |  |

(2) 创建文件备份执行方案



|       |           |
|-------|-----------|
| 方案类型  | 主机列表      |
| 方案名称  | host_list |
| 方案组   | 5 台主机     |
| 方案描述  |           |
| 执行时必填 | 是         |

| IPv4       | IPv6 | Agent 状态 | OS 名称        |  |
|------------|------|----------|--------------|--|
| 10.0.48.45 | ---  | 正常       | linux centos |  |
| 10.0.48.34 | ---  | 正常       | linux centos |  |
| 10.0.48.48 | ---  | 正常       | linux centos |  |
| 10.0.48.37 | ---  | 正常       | linux centos |  |
| 10.0.48.32 | ---  | 正常       | linux centos |  |

3. 创建执行方案

(1) 文件查询方案执行测试

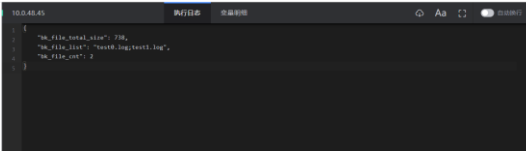
步骤执行详情 (伊蒙执行脚本\_20241205205316720)

状态: 执行成功 总耗时: 1.059s

步骤执行脚本\_20241205205316720

执行成功

| IP 组       | IPv6 组 | 耗时(s) | 策略组          | 返回码 |  |
|------------|--------|-------|--------------|-----|--|
| 10.0.48.45 | ---    | 0.200 | Default Area | 0   |  |
| 10.0.48.34 | ---    | 0.200 | Default Area | 0   |  |
| 10.0.48.48 | ---    | 0.201 | Default Area | 0   |  |
| 10.0.48.37 | ---    | 0.201 | Default Area | 0   |  |
| 10.0.48.32 | ---    | 0.100 | Default Area | 0   |  |



(2) 文件备份方案执行测试


步骤执行详情 (伊蒙执行脚本\_20241205210430628)

状态: 执行成功 总耗时: 1.066s

步骤执行脚本\_20241205210430628

执行成功

| IP 组       | IPv6 组 | 耗时(s) | 策略组          | 返回码 |  |
|------------|--------|-------|--------------|-----|--|
| 10.0.48.45 | ---    | 0.201 | Default Area | 0   |  |
| 10.0.48.34 | ---    | 0.200 | Default Area | 0   |  |
| 10.0.48.48 | ---    | 0.201 | Default Area | 0   |  |
| 10.0.48.37 | ---    | 0.201 | Default Area | 0   |  |
| 10.0.48.32 | ---    | 0.100 | Default Area | 0   |  |



(3) 记录执行方案 ID

| <input type="checkbox"/> | ID      | 执行方案名称          |
|--------------------------|---------|-----------------|
| <input type="checkbox"/> | 1000793 | 【21312450X】文件备份 |
| <input type="checkbox"/> | 1000791 | 【21312450X】文件查询 |

## 4. 后端实现文件查询接口

(1) 编写 `home_application/constants.py` 如下：

```
1  import os
2
3  # JOB执行作业的业务ID
4  JOB_BK_BIZ_ID = 3
5
6  # 作业执行结果查询的最大轮询次数
7  MAX_ATTEMPTS = 10
8
9  # 调用作业执行结果api的轮询间隔
10 JOB_RESULT_ATTEMPTS_INTERVAL = 0.2
11
12 # JOB作业平台HOST
13 BK_JOB_HOST = os.getenv("BKPAAS_JOB_URL")
14
15 # JOB 平台的状态码
16 WAITING_CODE = 2
17 SUCCESS_CODE = 3
18
19 # 默认HTTP状态码
20 WEB_SUCCESS_CODE = 0
21
22 # 作业方案ID
23 # SEARCH_FILE_PLAN_ID = 1000451 # 将这里的方案ID更改为你自己在JOB平台上新建的方案ID
24 # BACKUP_FILE_PLAN_ID = 1000452
25 SEARCH_FILE_PLAN_ID = 1000791
26 BACKUP_FILE_PLAN_ID = 1000793
```

(2) 在 `home_application/views.py` 中编写查询接口视图函数 `def search_file(request)`如下：

```
1  def search_file(request):
2      """
3      根据主机IP、文件目录和文件后缀，查询符合条件的主机文件
4      """
5
6      # 注意：先在constants.py中替换SEARCH_FILE_PLAN_ID为你自己在作业平台上新建的方案ID
7      host_id_list_str = request.GET.get("host_id_list")
8      host_id_list = [int(bk_host_id) for bk_host_id in host_id_list_str.split(",")]
9      kwargs = {
10         "bk_scope_type": "biz",
11         "bk_scope_id": JOB_BK_BIZ_ID,
12         "job_plan_id": SEARCH_FILE_PLAN_ID,
13         # TODO 修改为你创建的执行方案的全局变量
14         "global_var_list": [
15             {
16                 "name": "host_list",
17                 "server": {
18                     "host_id_list": host_id_list,
19                 },
20             },
21             {
22                 "name": "search_path",
23                 "value": request.GET.get("search_path"),
24             },
25             {
26                 "name": "suffix",
27                 "value": request.GET.get("suffix"),
28             },
29         ],
30     }
31
32     # 调用执行方案
33     client = get_client_by_request(request)
34     job_instance_id = client.jobv3.execute_job_plan(**kwargs).get("data").get("job_instance_id")
35
36     kwargs = {
37         "bk_scope_type": "biz",
38         "bk_scope_id": JOB_BK_BIZ_ID,
39         "job_instance_id": job_instance_id,
40     }
41
42     attempts = 0
```

```

43 while attempts < MAX_ATTEMPTS:
44     # 获取执行方案执行状态
45     step_instance_list = client.jobv3.get_job_instance_status(**kwargs).get("data").get("step_instance_list")
46     if step_instance_list[0].get("status") == WAITING_CODE:
47         time.sleep(JOB_RESULT_ATTEMPTS_INTERVAL)
48     elif step_instance_list[0].get("status") != SUCCESS_CODE:
49         res_data = {
50             "result": False,
51             "code": WEB_SUCCESS_CODE,
52             "message": "search failed",
53         }
54         return JsonResponse(res_data)
55     elif step_instance_list[0].get("status") == SUCCESS_CODE:
56         break
57     attempts += 1
58
59     step_instance_id = step_instance_list[0].get("step_instance_id")
60
61     log_list = []
62     for bk_host_id in host_id_list:
63         data = {
64             "bk_scope_type": "biz",
65             "bk_scope_id": JOB_BK_BIZ_ID,
66             "job_instance_id": job_instance_id,
67             "step_instance_id": step_instance_id,
68             "bk_host_id": bk_host_id,
69         }
70
71         # 查询执行日志
72         response = client.jobv3.get_job_instance_ip_log(**data).get("data")
73         step_res = response.get("log_content")
74         json_step_res = json.loads(step_res)
75
76         json_step_res["bk_host_id"] = response.get("bk_host_id")
77         log_list.append(json_step_res)
78
79     res_data = {
80         "result": True,
81         "code": WEB_SUCCESS_CODE,
82         "data": log_list,
83     }
84     return JsonResponse(res_data)
85

```

## 5. 后端实现文件备份接口

- (1) 在 `home_application/models.py` 中创建 `BackupRecord` 如下:

```

1 from django.db import models
2
3 class BackupRecord(models.Model):
4     bk_host_id = models.CharField(verbose_name="主机ID", max_length=1024)
5     bk_file_dir = models.CharField(verbose_name="备份目录", max_length=1024)
6     bk_file_suffix = models.CharField(verbose_name="文件名后缀", max_length=255)
7     bk_backup_name = models.CharField(verbose_name="备份文件名", max_length=1024)
8     bk_file_create_time = models.CharField(verbose_name="备份时间", max_length=30)
9     bk_file_operator = models.CharField(verbose_name="备份人", max_length=30)
10    bk_job_link = models.CharField(verbose_name="JOB 链接", max_length=100)
11
12    class Meta:
13        verbose_name = "备份记录"
14        verbose_name_plural = verbose_name

```

- (2) 使用 `python manage.py makemigrations` 与 `python manage.py migrate` 执行数据库迁移
- (3) 在 `home_application/views.py` 中编写文件备份接口函数与后端备份记录查询函数如下:



```

1 def backup_file(request):
2     """
3     根据主机IP、文件目录和文件后缀，备份符合条件的主机文件到指定目录
4     """
5
6     # 注意：先在constants.py中替换BACKUP_FILE_PLAN_ID为你自己在作业平台上新建的方案的ID
7     host_id_list_str = request.GET.get("host_id_list")
8     host_id_list = [int(bk_host_id) for bk_host_id in host_id_list_str.split(",")]
9     search_path = request.GET.get("search_path")
10    suffix = request.GET.get("suffix")
11    kwargs = {
12        "bk_scope_type": "biz",
13        "bk_scope_id": JOB_BK_BIZ_ID,
14        "job_plan_id": BACKUP_FILE_PLAN_ID,
15        # TODO 修改为你创建的执行方案的全局变量
16        "global_var_list": [
17            {
18                "name": "host_list",
19                "server": {
20                    "host_id_list": host_id_list,
21                },
22            },
23            {
24                "name": "search_path",
25                "value": search_path
26            },
27            {
28                "name": "suffix",
29                "value": suffix
30            },
31            {
32                "name": "backup_path",
33                "value": request.GET.get("backup_path"),
34            },
35        ],
36    }
37
38    # 调用执行方案
39    client = get_client_by_request(request)
40    job_instance_id = client.jobv3.execute_job_plan(**kwargs).get("data").get("job_instance_id")
41
42    kwargs = {
43        "bk_scope_type": "biz",
44        "bk_scope_id": JOB_BK_BIZ_ID,
45        "job_instance_id": job_instance_id,
46    }
47    attempts = 0
48    while attempts < MAX_ATTEMPTS:
49        # 获取执行方案执行状态
50        step_instance_list = client.jobv3.get_job_instance_status(**kwargs).get("data").get("step_instance_list")
51        if step_instance_list[0].get("status") == WAITING_CODE:
52            time.sleep(JOB_RESULT_ATTEMPTS_INTERVAL)
53        elif step_instance_list[0].get("status") != SUCCESS_CODE:
54            res_data = {
55                "result": False,
56                "code": WEB_SUCCESS_CODE,
57                "message": "backup failed",
58            }
59            return JsonResponse(res_data)
60        elif step_instance_list[0].get("status") == SUCCESS_CODE:
61            break
62        attempts += 1
63
64    step_instance_id = step_instance_list[0].get("step_instance_id")
65
66    for bk_host_id in host_id_list:
67        data = {
68            "bk_scope_type": "biz",
69            "bk_scope_id": JOB_BK_BIZ_ID,
70            "job_instance_id": job_instance_id,
71            "step_instance_id": step_instance_id,
72            "bk_host_id": bk_host_id,
73        }
74
75        # 查询执行日志
76        response = client.jobv3.get_job_instance_ip_log(**data).get("data")
77        step_res = response.get("log_content")
78        json_step_res = json.loads(step_res)
79
80        for step_res in json_step_res:
81            # 创建备份记录
82            step_res["bk_host_id"] = bk_host_id
83            step_res["bk_file_dir"] = search_path
84            step_res["bk_file_suffix"] = suffix
85            step_res["bk_file_operator"] = request.user.username
86            step_res["bk_job_link"] = "{}biz/{}/execute/task/{}".format(
87                BK_JOB_HOST,
88                JOB_BK_BIZ_ID,
89                job_instance_id,
90            )
91            BackupRecord.objects.create(**step_res)
92
93        res_data = {
94            "result": True,
95            "data": "success",
96            "code": WEB_SUCCESS_CODE,
97        }
98    return JsonResponse(res_data)
99
100
101 def get_backup_record(request):
102     """
103     查询备份记录
104     """
105     res_data = {
106         "result": True,
107         "data": list(BackupRecord.objects.all().order_by("-id").values()),
108         "code": WEB_SUCCESS_CODE,
109     }
110     return JsonResponse(res_data)

```

## 6. 添加文件查询、文件备份、文件备份查询到路由中

```
1  from django.conf.urls import url
2
3  from . import views
4
5  urlpatterns = (
6      url(r"^$", views.home),
7      url(r"^dev-guide/$", views.dev_guide),
8      url(r"^contact/$", views.contact),
9      url(r"biz-list", views.get_bizs_list),
10     url(r"set-list", views.get_sets_list),
11     url(r"module-list", views.get_modules_list),
12     url(r"host-list", views.get_hosts_list),
13     url(r"host-detail", views.get_host_detail)
14 )
15
```

## 7. 部署到蓝鲸平台

### (1) 添加环境变量

default frontend + ⚙️

构建配置 进程配置 环境变量 挂载卷 可观测性 增强服务 更多配置

① 环境变量可以用来改变应用在不同环境下的行为；除自定义环境变量外，平台也会写入内置环境变量。 [查看内置环境变量](#)

全部 所有环境 预发布环境 生产环境

| Key                 | Value                         | 生效环境  |
|---------------------|-------------------------------|-------|
| CORS_ALLOWED_ORIGIN | http://apps1.ce.bktencent.com | 预发布环境 |

default frontend + ⚙️

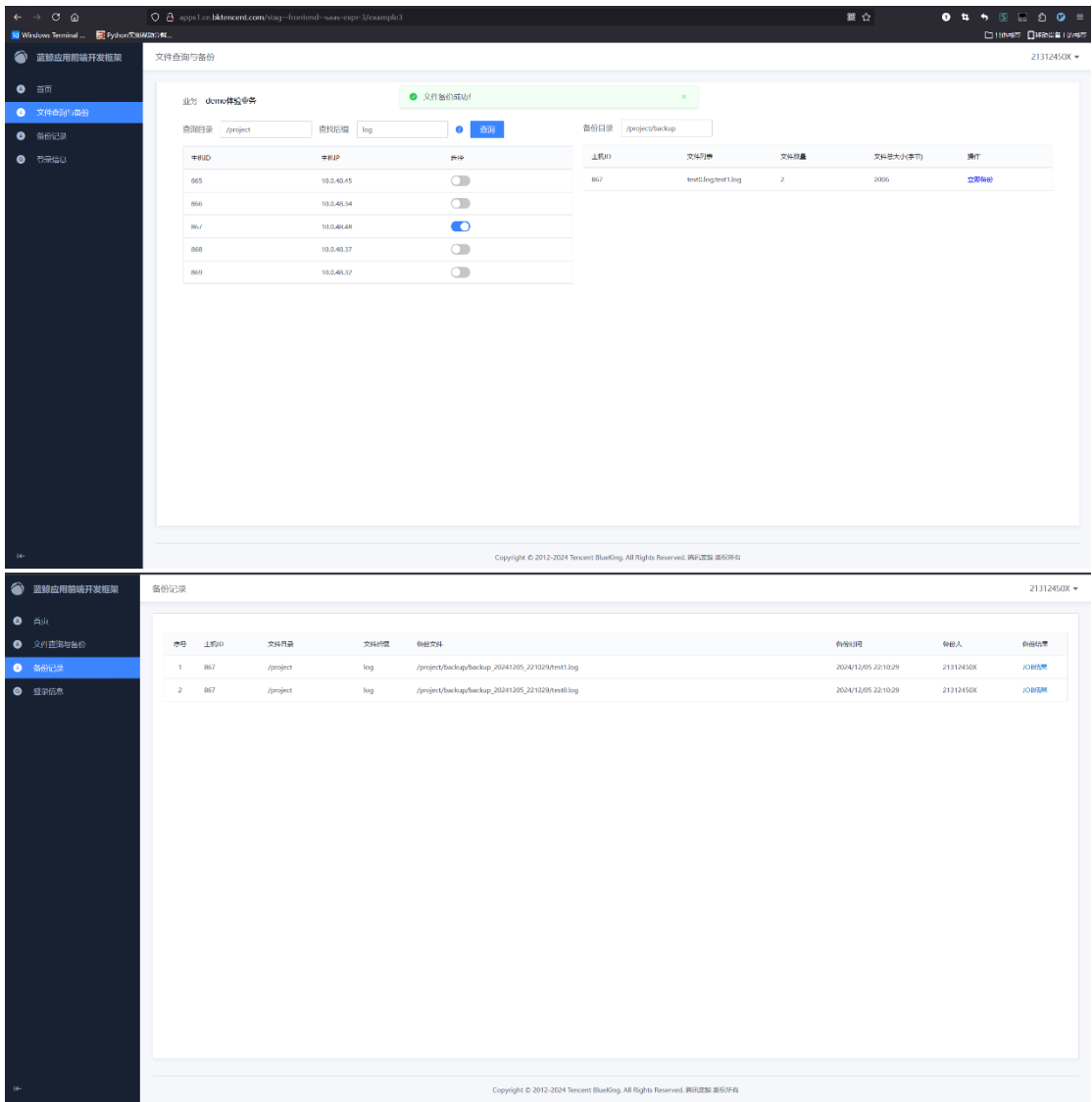
构建配置 进程配置 环境变量 挂载卷 可观测性 增强服务 更多配置

① 环境变量可以用来改变应用在不同环境下的行为；除自定义环境变量外，平台也会写入内置环境变量。 [查看内置环境变量](#)

全部 所有环境 预发布环境 生产环境

| Key                   | Value   | 生效环境  |
|-----------------------|---|-------|
| BK_BACKEND_API_PREFIX | https://apps1.ce.bktencent.com/stag-saas-expr-3 | 预发布环境 |

(2) 部署上线，最终效果如下：



## 六、 实验心得与体会

通过这次基于 JOB 作业平台实现游戏主机日志备份的实验，我深入了解了蓝鲸作业平台的功能和使用方法。编写脚本实现文件查询和备份，不仅提升了我的脚本编写能力，也让我掌握了作业平台接口的调用方法。在实现备份记录的建模和数据库操作过程中，我更加熟悉了 Django 的 ORM 操作。这次实验不仅巩固了我的 SaaS 开发技能，也让我体会到了自动化运维的强大功能。