蚂蚁集团

研发效能平台 LinkE

部署指南

产品版本：AntStack Plus 1.13.1

文档版本：20230710

蚂蚁集团
ANT GROUP

法律声明

蚂蚁集团版权所有©2022，并保留一切权利。

商标声明

免责声明

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.产品与应用介绍

LinkE 是一款基于 SOFA 技术栈的 Devops 平台，提供项目管理、代码托管、研发迭代、测试部署、等能力。研发效能平台对外交付部署主要基于云游进行相关操作，此文档用于云游上交付 LinkE 的部署指导参考。

LINKEBASE

LinkE 的基础服务，提供了元数据管理、配置管理、身份鉴权管理、日志收集、NoSQL 数据库。是整个研发效能平台的基础应用。

模块	说明
linkuredis	Redis 集群缓存服务，如有自建或底座 Redis 服务可不部署，纯缓存无持久化，不需要磁盘。
linkemongo	MongoDB 集群数据库，如有自建或底座 Mongo 服务可不部署，如部署则数据盘做 Mongo 数据存储需要有快照策略。
linkenexus	mvn 服务 Nexus 仓库，一般有自建不部署。
linku	LinkE 系统内部的统一用户服务。
linkm	LinkE 系统内部的统一元数据服务。
fabric	LinkE 系统的配置变更服务。
linklog	LinkE 的基础日志应用服务。

ANTCODE

蚂蚁的代码服务，提供了代码的存储，Git 服务与 Web 界面，并提供了基线管理功能。

模块	说明
antcoderedis	Redis 集群缓存服务，如有自建或底座 Redis 服务可不部署。
antcodenode	代码存储，数据盘做代码数据存储需要有快照策略。
batman-shard	代码存储分片应用服务。
batman-proxy	代码存储服务 Proxy。
antcodeweb	代码服务控制台 Web 应用。
codecenter	LinkE 内部统一的代码服务中心。

LINKB

构建服务，提供对代码打包和镜像构建的能力。

模块	说明
linkb	镜像构建的服务端。
linkb runner	镜像构建消费执行器应用。

LINKECI

持续集成与代码扫描能力，可以执行用户编排好的流水线并提供代码规约扫描与执行自动化测试的能力。

模块	说明
acijenkins	CI 所用的 Jenkins 集群。
aci	执行 CI 任务，调度 Jenkins 集群的应用。
aclinkelib	PMD、CI、构建、发布等所有组件的中心服务。
aclinkecore	流水线编排、执行的服务。
linka	代码分析 PMD、覆盖率等。

LINKEPORTAL

研发效能的门户，提供了统一的前端入口和部署、门户、流程、任务管理。

模块	说明
deploycore	研发环境，应用服务及发布单管理。
bahamut	工作台入口，负责研发迭代的应用。
linkflow	流程审批中心应用。
linkt	项目管理、工作项缺陷需求管理。
linkeonex	LinkE 的前端界面应用入口和菜单管理。

2.应用依赖关系

2.1. 数据库依赖

2.1.1. MySQL 依赖

> 说明
>
> 推荐使用 MySQL 5.7 版本。

在 LinkE 目前版本中，MySQL 仅能使用默认 3306 端口提供服务，因为部分应用模块集成了蚂蚁 zdal 数据库连接中间件，不需要非标端口的 MySQL 。MySQL 需要开启的配置项如下：

- **全局索引配置项**

  启用 `innodb_large_prefix` ，同时指定 `innodb_file_format=barracuda` ， `innodb_file_per_table=true` ， `innodb_default_row_format=DYNAMIC` 。
- **参数 sql_mode 配置项**
- **参数 lower_case_table_names 配置**

2.1.2. 对象存储依赖

支持阿里云 OSS 与 Amazon S3 对象存储协议。

> 说明
>
> - 阿里云 OSS 支持公私有云版本。
> - Amazon S3 协议应用比较广泛，通常私有云环境使用云游搭建的 minio 服务。
> - 阿里云公有云 OSS 服务也支持使用 S3 协议访问，但私有云版本不一定支持。

使用用对象存储服务保存 LinkE 生成的或运行必需的文件，包括：

- 执行 mvn 编译时所需 nexus 仓库 settings.xml 配置

  默认 bucket 为 `bahamutstorage-bahamutstorage`
- 经典应用服务编译生成的产物包文件

  默认 bucket 为 `bahamutstorage-bahamutstorage`
- 执行编译构建、单元测试等功能的运行日志文件

  默认 bucket 为 `logstore-logstore`
- 项目协作 linkt 模块与流程中心 linkflow 模块的附件上传功能

  默认 bucket 为 `linkestorage-linkestorage`
- 代码服务模块文件上传功能默认 bucket 为 `antcodeoss-antcodeoss`

在 LinkE 的应用中，全局 OSS 的认证配置可通过云游产品公共参数中填入，如下图所示配置项，这里的 OSS 代表对象存储：

渲染参数 | 确认全部 | 应用启动参数 | 全局参数 | 产品公共参数 | 编辑视图 | oss

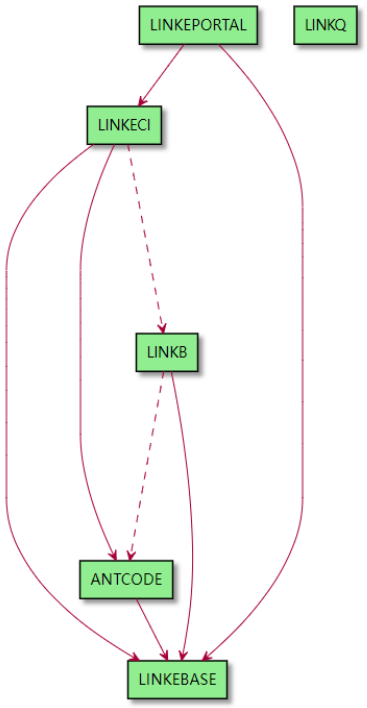
	产品码	键	值模板	渲染结果	描述	操作
已确认	LINKEBASE	OSS_ACCESS_KEY	LTAI5tHQFxsE885jyipnPdpA	minio	oss公用accessKey	编辑
已确认	LINKEBASE	OSS_ACCESS_SECRET	敏感 [redacted] yU	[redacted]	oss密钥	编辑
已确认	LINKEBASE	OSS_ENDPOINT	oss-cn-shanghai.aliyuncs.com	http://100.81.68.18:9000	公用oss地址	编辑
已确认	LINKEPORTAL	OSS_ACCESS_KEY	minio	minio	oss公用accessKey	编辑
已确认	LINKEPORTAL	OSS_ACCESS_SECRET	敏感 [redacted]	oss密钥	编辑	
已确认	LINKEPORTAL	OSS_ENDPOINT	http://100.81.68.18:9000	http://100.81.68.18:9000	公用 oss endpoint 地址	编辑

在应用的启动参数里，可修改对象存储的类型与 Bucket 的配置声明：

表达式引用	LINKEPORTAL	bahamutcloud	ANTCLOUD_SOFA_ossBucketName	#if (${prod.LINKEPORTAL.DEPLOY_TYPE} == 'public') oss-engineer-efficiency-prod-public #elseif (${res.LINKEPORTAL.storage.bahamutstorage.exist}) ${res.LINKEPORTAL.storage.bahamutstorage.bucketName} #else bahamutstorage-bahamutstorage #end	bahamutstorage-bahamutstorage	编辑
表达式引用	LINKEPORTAL	bahamutcloud	ANTCLOUD_SOFA_ossEndpoint	#if (${prod.LINKEPORTAL.DEPLOY_TYPE} == 'public') cn-hangzhou.alipay-pub.aliyun-inc.com #elseif (${res.LINKEPORTAL.storage.bahamutstorage.exist}) ${res.LINKEPORTAL.storage.bahamutstorage.privateEndpoint} #else ${prod.LINKEPORTAL.OSS_ENDPOINT} #end	http://100.81.68.18:9000	编辑
表达式引用	LINKEPORTAL	bahamutcloud	ANTCLOUD_SOFA_ossType	${prod.LINKEPORTAL.STORAGE_TYPE}	s3	编辑

2.2. LinkE 内部依赖

LinkE 内部依赖



> 说明
>
> 上图中实线表示强依赖，虚线表示弱依赖。

2.3. 外部依赖

- **OP**

  金融云网关，对接金融云操作都是通过 OP 完成的。

- **IAM**

  金融云身份鉴权，专有云 LinkE 是对接的金融云的用户体系。

- **ONECONSOLE**

  onex 应用，进行前端的应用请求转发。

- **产品集 CAFE Standard & CAFE Classic**

  LinkE 有一大重要功能就是对接 PaaS，需要有 PaaS 在才能完成整个 Devops 生命周期。

- **产品集共享中间件（弱依赖）**

  LinkE 中提供了中间件配置变更的能力，通过 OP > OSP > 各个中间件的 OpenAPI 完成配置同步与生效。

- **APP_COMPOSE（弱依赖）**

  应用编排，用来创建带前后置任务和中间件生效的 deps 发布单（经典应用类型）

2.4. 其他依赖

云游 LOCAL

目前部署都是通过云游的编排能力进行部署。

底座资源

- ECS/ACS：服务器资源。
- RDS/MYSQL：数据库资源。
- ALB/SLB：负载均衡资源。
- ADNS：提供域名进行访问。
- OSS/S3：产物与日志存储。

3.部署准备

3.1. 部署资源准备

3.1.1. 应用容器资源

以下为 Global 产品集中的资源单机房部署默认规格列表，如需扩缩容数量请在解决方案中调整应用容器个数。

产品	应用	服务器类型	CPU	内存	硬盘	数量（台）
LINKEBASE	linkuredis	ECS	1C	2G	100G	2
	linkemongo	ECS	2C	4G	500G	3
	linkenexus	ECS	4C	8G	1000G	0
	linku	ECS	2C	4G	100G	2
	linkm	ECS	2C	4G	100G	2
	fabric	ECS	2C	4G	100G	2
	linklog	ECS	2C	4G	100G	2
ANTCODE	antcoderedis	ECS	1C	2G	100G	2
	antcodenode	ECS	2C	4G	1000G	2
	batman-shard	ECS	2C	4G	100G	2
	batman-proxy	ECS	2C	4G	100G	2
	antcodeweb	ECS	2C	4G	100G	2
	codecenter	ECS	2C	4G	100G	2
LINKB	linkb	ECS	2C	4G	100G	2
	linkb runner	ECS	4C	8G	200G	2

LINKECI	acijenkins	ECS	4C	8G	200G	2
	aci	ECS	2C	4G	100G	2
	aclinkelib	ECS	2C	4G	100G	2
	aclinkecore	ECS	2C	4G	100G	2
	linka	ECS	2C	4G	100G	2
LINKEPORTAL	deploycore	ECS	2C	4G	100G	2
	bahamut	ECS	2C	4G	100G	2
	linkflow	ECS	2C	4G	100G	2
	linkt	ECS	2C	4G	100G	2
	linkeonex	ECS	2C	4G	100G	2

物理资源占用汇总

CPU规格	内存规格	磁盘规格	实例数量	CPU	内存	磁盘
4	8	200	4	16	32	800
3	6	100	2	6	12	200
2	4	100	32	64	124	3200
1	2	100	4	4	8	400
1	1	100	2	2	2	200
基础部署包				92	178	

3.1.2. 存储资源

LinkE 中多个应用会用到对象存储，需要一组存储的 Endpoint 和 AK、SK 配置和多个存储空间，导入解决方案后可在解决方案中查看存储资源，按照云游的命名规范，Bucket 的名称为 `资源名称-资源名称` 。

LinkE 相应的存储空间名列表为：

- logstore-logstore
- bahamutstorage-bahamutstorage
- antcode_oss-antcode_oss
- linkestorage-linkestorage

如果云游 Local 不支持自动创建存储空间，则需要手动创建好存储空间后录入云游 Local 的数据库中，创建好 LinkE 所有产品所使用的存储空间如下。

> 说明
>
> 专有云内网环境一般会选择创建公共读写，也可以创建私有读写保证后面录入的 AK 和 SK 密钥正确即可

- 阿里云底座：在 ASC 阿里云控制台对象存储产品中创建。
- AntStack 底座：通过 afsserver 的 create_bucket 接口来创建。
- AntStack Plus 底座：通过 minio 的控制台创建。

> 重要
>
> minio 的存储空间名称不能使用下划线，因此如果使用 minio 手动创建时需要替换 `antcode_oss` 为 `antcodeoss` 。

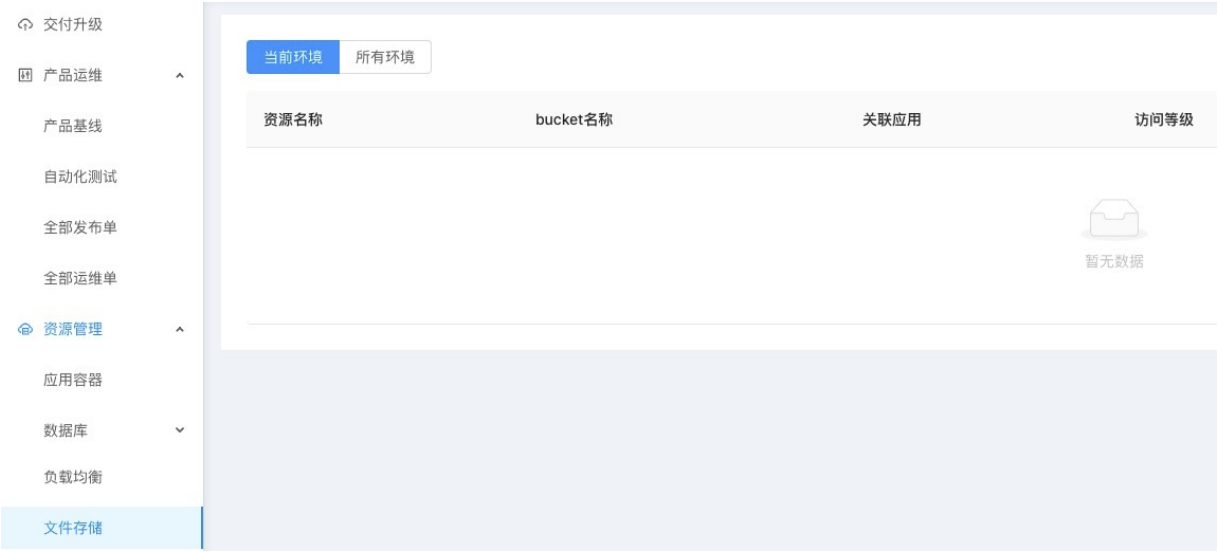
录入存储资源

如果需要在云游上查看存储资源并使用云游渲染存储参数，则需要进行云游数据库录入操作。

```
LINKEBASE -- logstore
privateEndpoint: xxx.xxx.xxx/
publicEndpoint: http://xxx.xxx.xxx/
bucketName: logstore-logstore
accessKey:
accessSecret:

LINKEPORTAL -- bahamutstorage
privateEndpoint: xxx.xxx.xxx/
publicEndpoint: http://xxx.xxx.xxx/
bucketName: bahamutstorage-bahamutstorage
accessKey:
accessSecret:

LINKEPORTAL -- antcode_oss
privateEndpoint: xxx.xxx.xxx/
publicEndpoint: http://xxx.xxx.xxx/
bucketName: antcode_oss-antcode_oss
accessKey:
accessSecret:

LINKEPORTAL -- linkestorage
privateEndpoint: xxx.xxx.xxx/
publicEndpoint: http://xxx.xxx.xxx/
bucketName: linkestorage-linkestorage
accessKey:
accessSecret:
```

> **说明**
>
> AK 和 SK 可由现场交付同学补充。
>
> - 阿里云底座从 OSS 控制台可获取。
> - AntStack Plus 底座获取 minio 的 AK 和 SK。

录入存储资源之后如果在云游的文件存储菜单中查看不到已创建的 Bucket 信息或者查看文件存储报错，如下图所示：



此时需要查看云游 Local 的环境设置，找到系统配置，如果界面上没有系统配置则将 URL 中 `settings` 改为 `system-settings` 进行访问，见下图：

增加 Local 中的环境调用存储的配置：

```
{
  "region":"cn-hohhot-ste3-d01",
  "endpointPublic":"xxx.xxx.xxx",
  "endpointPrivate":"xxx.xxx.xxx",
  "accessKey":"",
  "accessKeySecret":"",
}
```

key 为 `storage_conf` ：

3.1.3. 数据库资源

数据库名称（schema）	数据库用户名	连接密码	被引用应用
linku_cloud	linke	默认值	linkucloud

fabric_cloud	linke	默认值	linkmcloud
metacenter_cloud	linke	默认值	linkmcloud
linklog	linke	默认值	linklog
antcode	linke	默认值	antcodeweb
batmanshard	linke	默认值	batmanshard, batmanproxy, antcodenode
codecenter	linke	默认值	mycodecenter
linkb_db_01	linke	默认值	antbuild
core_cloud	linke	默认值	aclinkecore
linkt_db	linke	默认值	linktcloud
deploycore_db	linke	默认值	deploycore
linkflow_db	linke	默认值	linkflowcloud
linkgw_cloud	linke	默认值	bahamut, antcodeweb, linklog, linkt, linkflow

自动创建：一般情况下，数据库资源由云游管理，如 Local 环境支持自动创建 Schema，则选择自动创建即可。

手动创建：手动创建数据库实例时一般选择共享的物理实例进行添加 Schema，此时则不需要区分默认的物理实例，均使用同一个物理实例进行 Schema 的创建即可（如区分请按照上述数据库列表的默认物理实例创建和添加物理实例），按照如下语句创建数据库及用户并授权（需要用管理员账号登录到 Local 环境中所使用的数据库物理实例中）：

```
create schema <db_name>;
create user '[用户名称]'@'%' identified by '[用户密码]';
grant all privileges on `[用户名称]`.* TO '[用户名称]'@'%';
```

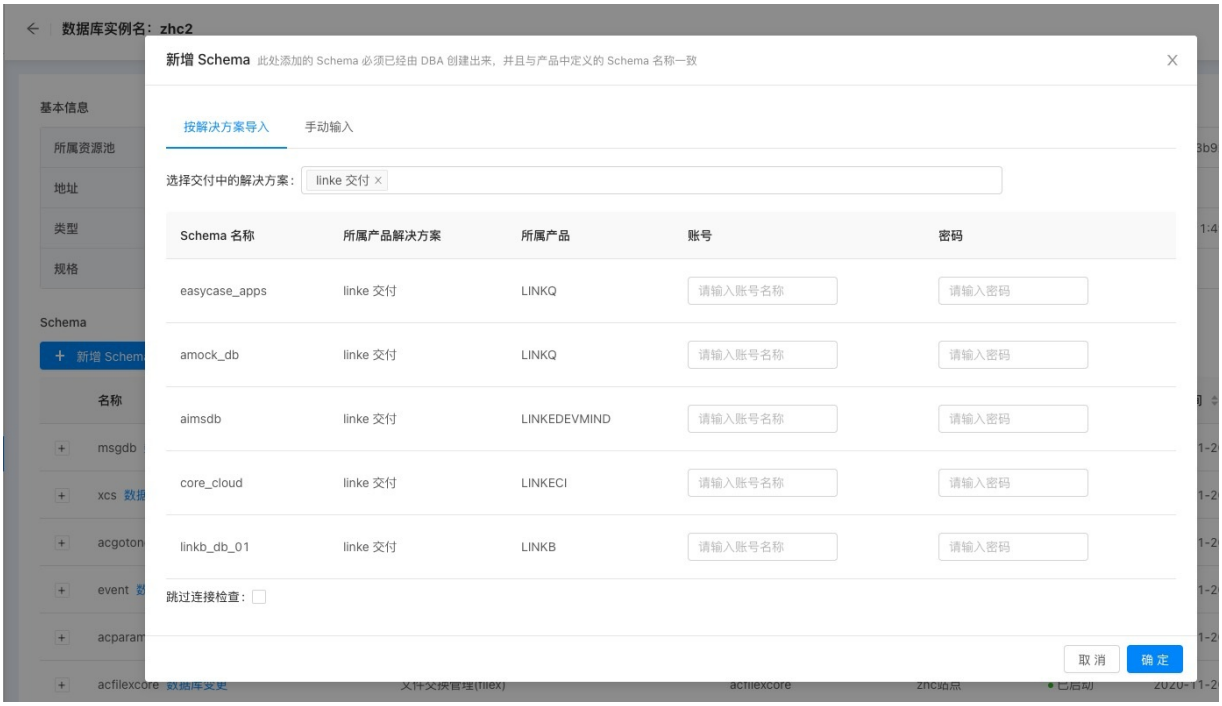
具体的创建 SQL 如下：

```
create schema linkgw_cloud;
create schema linku_cloud;
create schema fabric_cloud;
create schema metacenter_cloud;
create schema linklog;
create schema antcode;
create schema batmanshard;
create schema codecenter;
create schema linkb_db_01;
create schema core_cloud;
create schema linkt_db;
create schema deploycore_db;
create schema linkflow_db;

-- 部署方式1：使用统一的数据库账号访问 linke 数据库
create user 'linke'@'%' identified by 'PaaS123456';

grant all privileges on `linke`.* TO 'linkgw_cloud'@'%';
grant all privileges on `linke`.* TO 'fabric_cloud'@'%';
grant all privileges on `linke`.* TO 'linku_cloud'@'%';
grant all privileges on `linke`.* TO 'metacenter_cloud'@'%';
grant all privileges on `linke`.* TO 'linklog'@'%';
grant all privileges on `linke`.* TO 'antcode'@'%';
grant all privileges on `linke`.* TO 'batmanshard'@'%';
grant all privileges on `linke`.* TO 'codecenter'@'%';
grant all privileges on `linke`.* TO 'linkb_db_01'@'%';
grant all privileges on `linke`.* TO 'core_cloud'@'%';
grant all privileges on `linke`.* TO 'linkt_db'@'%';
grant all privileges on `linke`.* TO 'deploycore_db'@'%';
grant all privileges on `linke`.* TO 'linkflow_db'@'%';

-- 如果数据库是 mysql 还需要为 linkflow_db 执行编码配置，设置编码为 utf8mb4
-- oceanbase 数据库则不需要，因为 utf8mb4 是 mysql 的专属编码格式
alter database linkflow_db character set utf8mb4;

-- 部署完成后，linkflow_db 的表也需要设置编码为 utf8mb4
alter table ACT_FO_FORM_DEFINITION character set utf8mb4;
alter table ACT_FO_FORM_DEFINITION modify NAME_ varchar(50) character set utf8mb4;
```

云游上新增 Schema

在 Local 环境中进入物理实例按照解决方案导入 Schema，并设置账号密码。

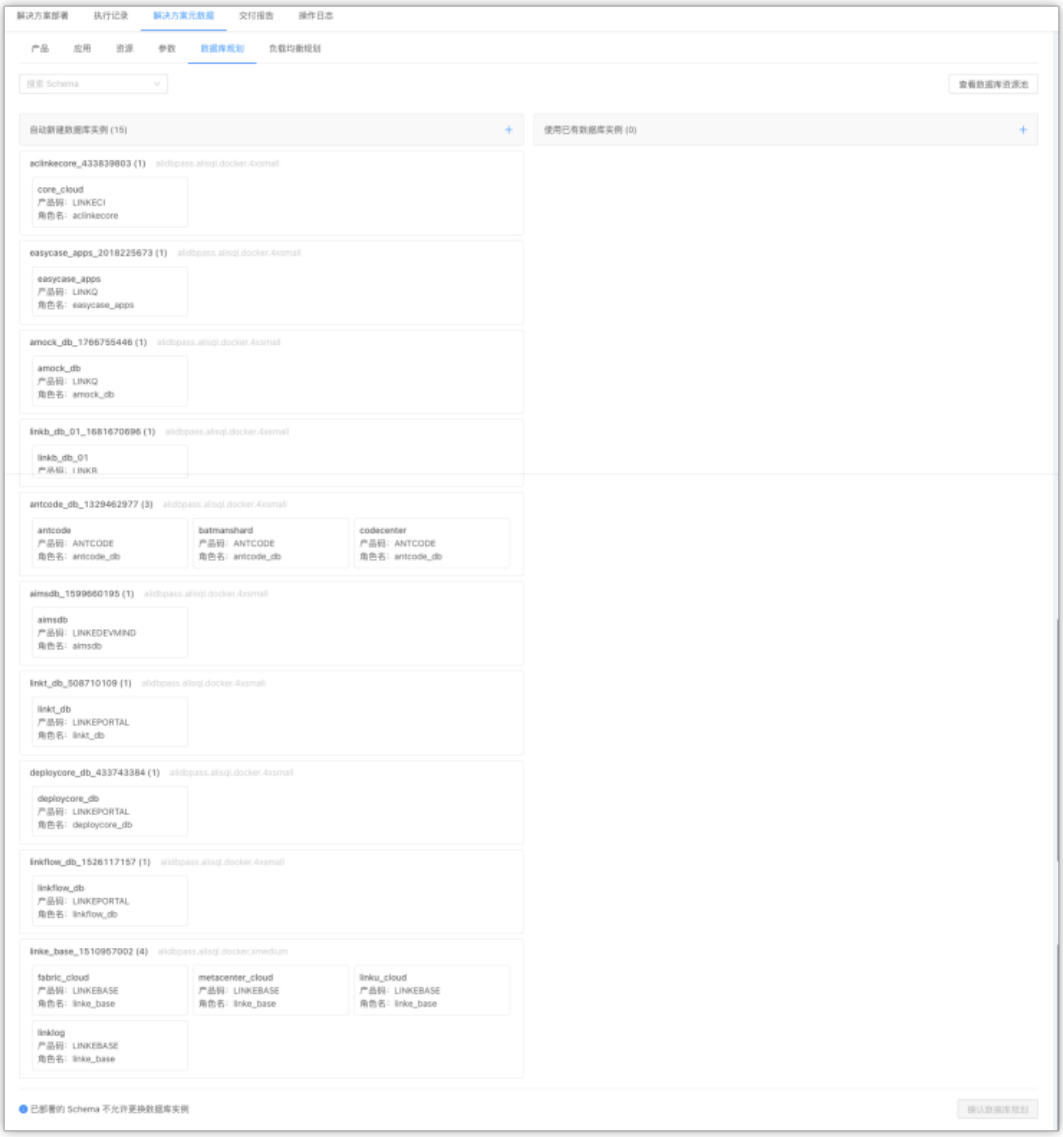
> 说明
>
> 账号和 Schema 名称一致，密码按照创建账号时使用的密码，默认均使用 LinkE 默认密码：`PaaS123456`。



规划数据库

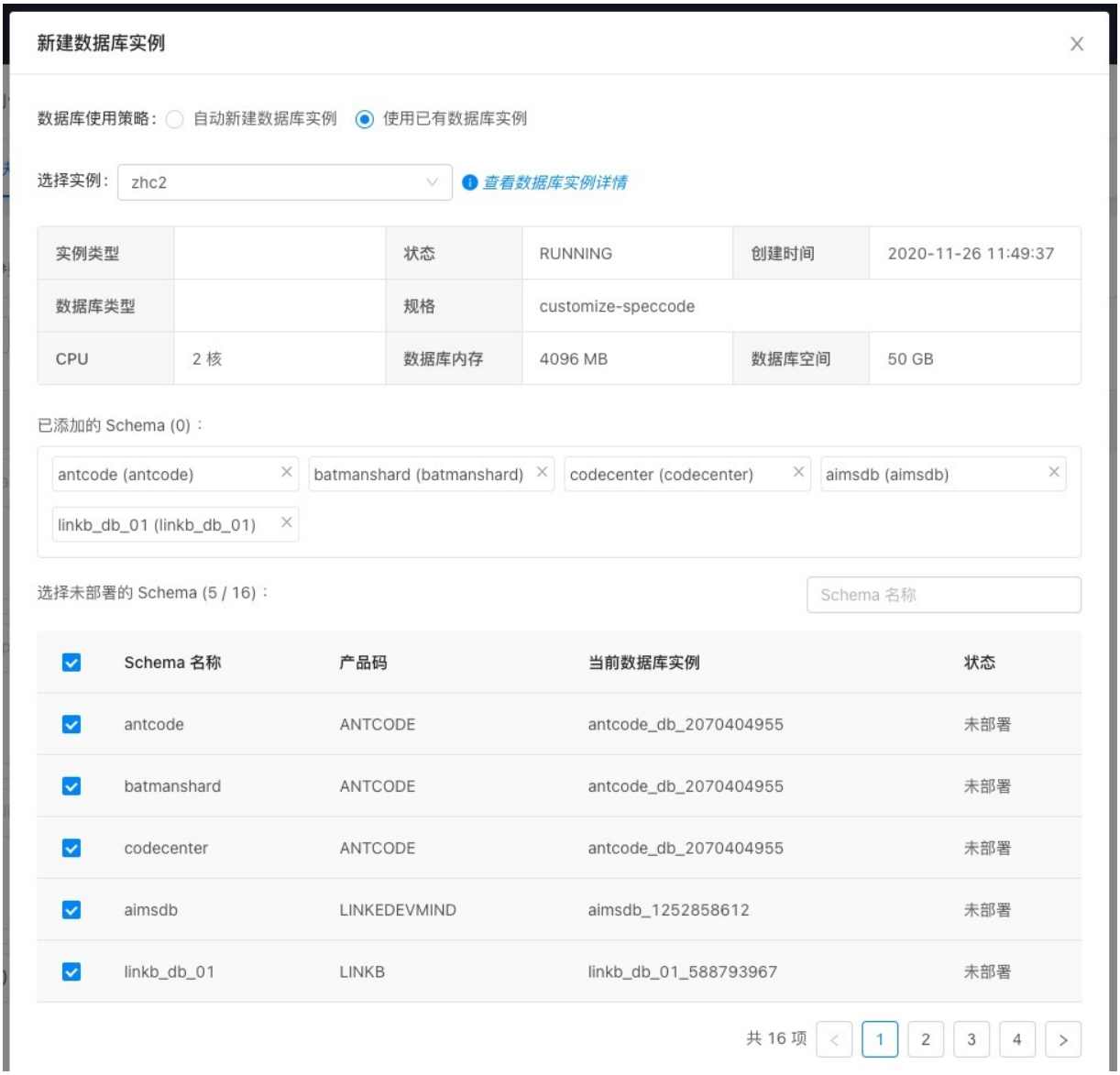
根据 Local 环境选择新建实例还是复用实例，自动创建数据的情况直接选择新建，复用实例则添加已创建的已有的实例。

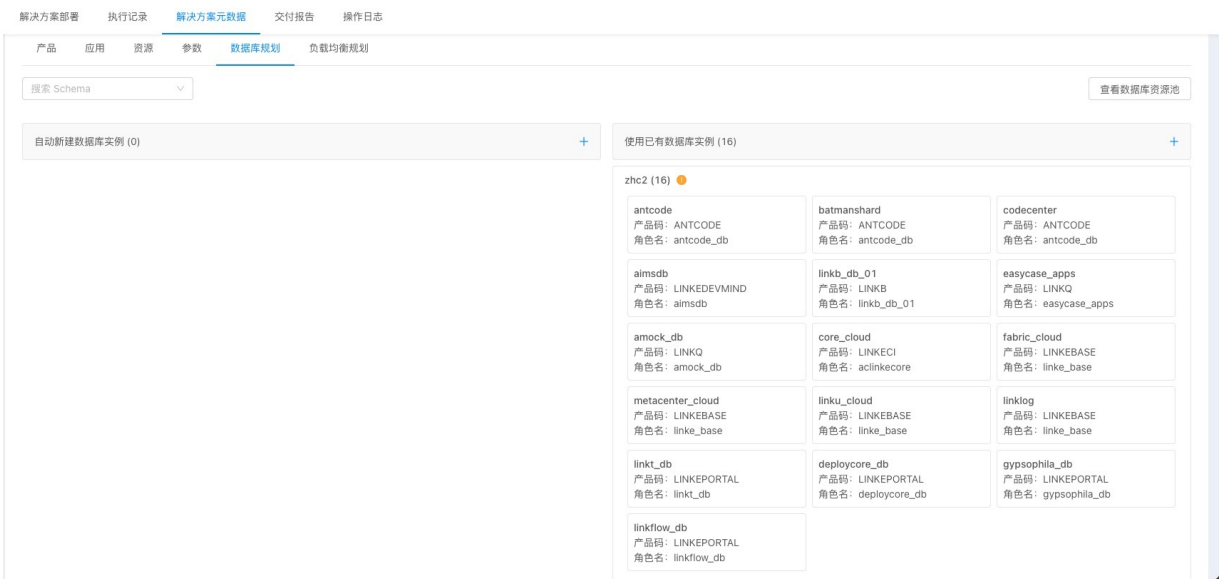
选择自动新建数据库实例



复用已有实例时选择已创建的 Schema 实例，并将所有数据库添加进去

一般情况下手动创建 Schema 选择同一个共享的物理实例即可，所以添加时从一个实例中即可添加所有创建的 Schema，如果区分了各个产品而创建了默认物理实例，则分开进行添加。


新建数据库实例
数据库使用策略：自动新建数据库实例 使用已有数据库实例
选择实例：zhc2
查看数据库实例详情
实例类型
状态
RUNNING
创建时间
2020-11-26 11:49:37
数据库类型
规格
customize-speccode
CPU
2 核
数据库内存
4096 MB
数据库空间
50 GB
已添加的 Schema (0)：
antcode (antcode)
batmanshard (batmanshard)
codecenter (codecenter)
aimsdb (aimsdb)
linkb_db_01 (linkb_db_01)
选择未部署的 Schema (5 / 16)：
Schema 名称
Schema 名称
产品码
当前数据库实例
状态
antcode
ANTCODE
antcode_db_2070404955
未部署
batmanshard
ANTCODE
antcode_db_2070404955
未部署
codecenter
ANTCODE
antcode_db_2070404955
未部署
aimsdb
LINKEDEVMIND
aimsdb_1252858612
未部署
linkb_db_01
LINKB
linkb_db_01_588793967
未部署
共 16 项
1
2
3
4


解决方案部署
执行记录
解决方案元数据
交付报告
操作日志
产品
应用
资源
参数
数据库规划
负载均衡规划
搜索 Schema
查看数据库资源池
自动新建数据库实例 (0)
使用已有数据库实例 (16)
zhc2 (16)
antcode
产品码：ANTCODE
角色名：antcode_db
batmanshard
产品码：ANTCODE
角色名：antcode_db
codecenter
产品码：ANTCODE
角色名：antcode_db
aimsdb
产品码：LINKEDEVMIND
角色名：aimsdb
linkb_db_01
产品码：LINKB
角色名：linkb_db_01
easycase_apps
产品码：LINKQ
角色名：easycase_apps
amock_db
产品码：LINKQ
角色名：amock_db
core_cloud
产品码：LINKECI
角色名：aclinkecore
fabric_cloud
产品码：LINKEBASE
角色名：linke_base
metacenter_cloud
产品码：LINKEBASE
角色名：linke_base
linku_cloud
产品码：LINKEBASE
角色名：linke_base
linklog
产品码：LINKEBASE
角色名：linke_base
linkt_db
产品码：LINKEPORTAL
角色名：linkt_db
deploycore_db
产品码：LINKEPORTAL
角色名：deploycore_db
gypsophila_db
产品码：LINKEPORTAL
角色名：gypsophila_db
linkflow_db
产品码：LINKEPORTAL
角色名：linkflow_db

3.1.4. 负载均衡和域名

LinkE 所需要的负载均衡能力包括七层协议和四层协议，既有 HTTP 端口服务，也有其他非 HTTP 协议的端口服务。

需要公开访问域名的 SLB 如下：

```
# 必须的域名配置
linkconsole.${env.info.domain}  #解析到 oneconsole（负载均衡）或 oneconsole 容器
linkeapi.${env.info.domain}  #解析到 minionex（负载均衡）或 minionex 容器
code.${env.info.domain}  #解析到 antcode_slb 或 antcodeweb 容器
```

> 说明
>
> 未部署的应用则无需添加域名解析。

3.1.5. SidecarSet

暂无配置。

3.2. 确认宿主机内核参数配置

容器 buildrunner 需要依赖宿主机能力进行镜像构建（源自 buildkitd 工具的 rootless 部署方案）。实质上只有部署 buildrunner 容器的宿主机对此内核参数有依赖，但限制容器的部署节点（即为 node 添加 selector）操作对现场交付实施人员要求较高，故推荐将内核参数预置在节点装机模板。

操作步骤

在 k8s 集群上所有宿主机上，编辑 `/etc/sysctl.conf` 文件，尾部添加 `user.max_user_namespaces=28633`，然后执行 `sysctl -p` 生效内核参数配置。

> 说明
>
> 这是内核虚拟化功能参数，28633 这个数字不是强要求的，如 30000 也可以，确保这个数值大于 20000 即可。

脚本如下所示：

```
echo "user.max_user_namespaces=28633" >> /etc/sysctl.conf && sysctl -p
```

注意事项

```
[root@slave006 /root]
#echo 28633 > /proc/sys/user/max_user_namespaces
-bash: /proc/sys/user/max_user_namespaces: No such file or directory
```

如果执行 `sysctl -p` 如上图提示错误，那么 LinkE 的 buildrunner 程序将不能使用标准方案部署。

可改为 docker 构建方案，需要宿主机安装 dockerd 程序并将 `/var/run/docker.sock` 的权限设置为 `666`。然后 Pod 在启动时将把 docker.sock 文件挂载进容器，镜像构建功能借助 docker in docker 方案实现。

- **受影响的功能**：镜像构建功能，需要特殊处理。
- **通常原因**：OS系统版本较低（低于 centos7.6，或者使用低版本 alios7）。

3.3. 确认数据库配置

RDS MySQL 配置项

> 说明
>
> - 推荐直接使用 RDS MySQL 5.7 版本，使用 5.6 版本会遇到各类非标问题。
> - OceanBase 一般默认配置即可满足要求。

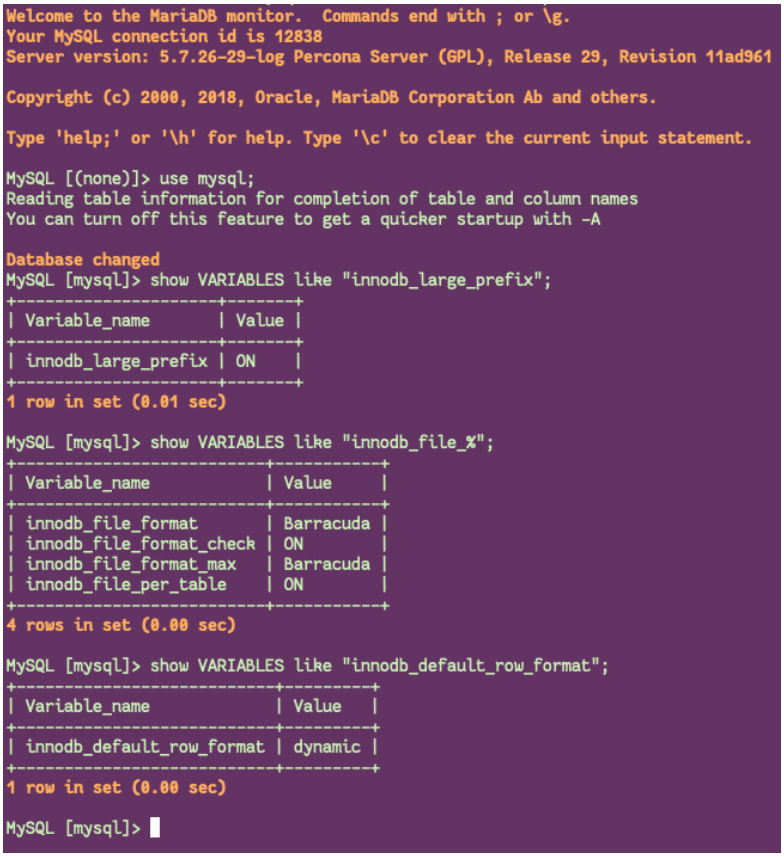
全局索引配置项

启用 `innodb_large_prefix`，同时指定 `innodb_file_format=barracuda`，`innodb_file_per_table=true`，`innodb_default_row_format=DYNAMIC`。

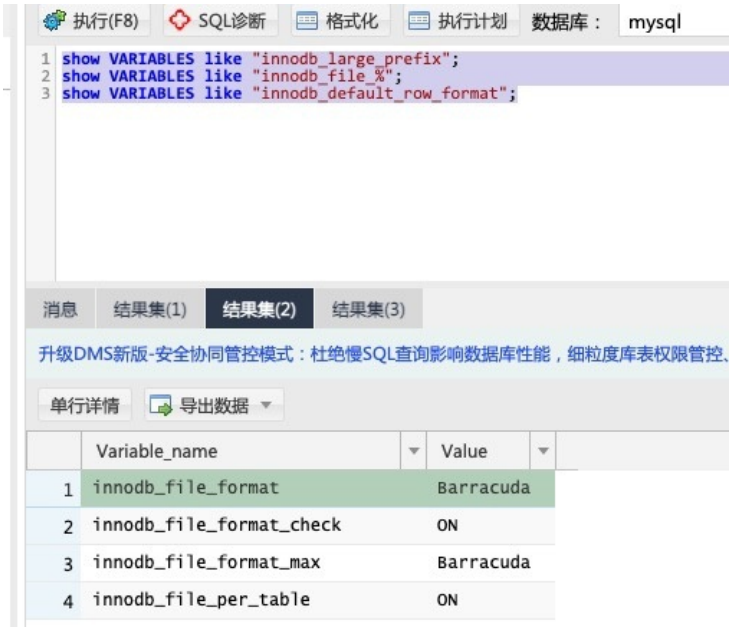
- **查看方法（执行 SQL）：**

```
show VARIABLES like "innodb_large_prefix";
show VARIABLES like "innodb_file_%";
show VARIABLES like "innodb_default_row_format";
```

- **结果示例：**



```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 12838
Server version: 5.7.26-29-log Percona Server (GPL), Release 29, Revision 11ad961

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MySQL [mysql]> show VARIABLES like "innodb_large_prefix";
+----------------------+-------+
| Variable_name        | Value |
+----------------------+-------+
| innodb_large_prefix | ON    |
+----------------------+-------+
1 row in set (0.01 sec)

MySQL [mysql]> show VARIABLES like "innodb_file_%";
+--------------------------+-----------+
| Variable_name            | Value     |
+--------------------------+-----------+
| innodb_file_format       | Barracuda |
| innodb_file_format_check | ON        |
| innodb_file_format_max   | Barracuda |
| innodb_file_per_table    | ON        |
+--------------------------+-----------+
4 rows in set (0.00 sec)

MySQL [mysql]> show VARIABLES like "innodb_default_row_format";
+---------------------------+---------+
| Variable_name             | Value   |
+---------------------------+---------+
| innodb_default_row_format | dynamic |
+---------------------------+---------+
1 row in set (0.00 sec)

MySQL [mysql]>
```



- **修改方法**

  如果拥有 MySQL 高权限账号，直接使用 `set global variable` 命令修改；如果使用了阿里云 RDS 服务（通常无高权限账号），可在 RDS 控制台上提交参数变更。

  - 针对 MySQL 5.7

    通常只需要修改全局参数 `innodb_large_prefix`，其他参数启动默认值通常已满足需求。

  - 针对 MySQL 5.6

    上述全局参数均需要修改，默认值不满足需求（因此推荐直接使用 MySQL 5.7 部署）。

- **影响**

  若不配置参数，强行部署时数据库变更流程无法走通，如下图所示：

```
`build_config_version` varchar(128) NULL COMMENT '构建版本',
`operator` varchar(32) NULL COMMENT '操作人',
`extra_info` longtext NULL COMMENT '扩展信息，json格式',
PRIMARY KEY (`id`),
UNIQUE `uk_index` (`app_name`, `tenant_id`, `build_apply_id`),
KEY `index` (`tenant_id`, `project_id`, `commit_id`)
) ENGINE = InnoDB AUTO_INCREMENT = 1 CHARSET = utf8mb4 COMMENT = '构建产物记录表',] Cause: java.lang.RuntimeException:
Execute sql error,com.mysql.jdbc.exceptions.jdbc4.MySQLSyntaxErrorException: Specified key was too long; max key length is
767 bytes, migrationId: SQL_202109092023230001,
at cn.com.antcloud.common.exception.inner.CommonInnerException.build2(CommonInnerException.java:123)
at cn.com.antcloud.common.exception.inner.CommonInnerException.build(CommonInnerException.java:113)
at com.alipay.cloud.apyunqing.common.util.ValidateExceptionWrapper.exception(ValidateExceptionWrapper.java:491)
at com.alipay.yunyou.biz.service.handler.rdb.DbSqlMigrationExecHandler.doExecMigration(DbSqlMigrationExecHandler.java:283)
at com.alipay.yunyou.biz.service.handler.rdb.DbSqlMigrationExecHandler.lambda$execute$2(DbSqlMigrationExecHandler.java:209)
at org.springframework.transaction.support.TransactionTemplate.execute(TransactionTemplate.java:133)
at com.alipay.yunyou.biz.service.handler.rdb.DbSqlMigrationExecHandler.execute(DbSqlMigrationExecHandler.java:206)
at com.alipay.yunyou.biz.service.handler.rdb.DbSqlMigrationExecHandler.execute(DbSqlMigrationExecHandler.java:60)
at com.alipay.yunyou.biz.service.engine.diffrun.DiffAndRunEngine.execute(DiffAndRunEngine.java:122)
at com.alipay.yunyou.biz.service.engine.diffrun.DiffAndRunEngine.execute(DiffAndRunEngine.java:68)
at com.alipay.yunyou.biz.service.cmd.ops.execute.engine.HandlerEngine2.doInvoke(HandlerEngine2.java:93)
at com.alipay.yunyou.biz.service.cmd.ops.execute.engine.HandlerEngine2.execute(HandlerEngine2.java:68)
at com.alipay.yunyou.biz.service.cmd.ops.execute.callback.ActionHandlerDelegate2.execute(ActionHandlerDelegate2.java:44)
at org.activiti.engine.impl.delegate.invocation.JavaDelegateInvocation.invoke(JavaDelegateInvocation.java:34)
at org.activiti.engine.impl.delegate.invocation.DelegateInvocation.proceed(DelegateInvocation.java:35)
at org.activiti.engine.impl.delegate.invocation.DefaultDelegateInterceptor.handleInvocation
(DefaultDelegateInterceptor.java:25)
at org.activiti.engine.impl.bpmn.behavior.ServiceTaskJavaDelegateActivityBehavior.execute
(ServiceTaskJavaDelegateActivityBehavior.java:40)
at org.activiti.engine.impl.bpmn.helper.ClassDelegate.execute(ClassDelegate.java:221)
at org.activiti.engine.impl.agenda.ContinueProcessOperation.executeActivityBehavior(ContinueProcessOperation.java:211)
at org.activiti.engine.impl.agenda.ContinueProcessOperation.executeSynchronous(ContinueProcessOperation.java:145)
at org.activiti.engine.impl.agenda.ContinueProcessOperation.continueThroughFlowNode(ContinueProcessOperation.java:102)
at org.activiti.engine.impl.agenda.ContinueProcessOperation.run(ContinueProcessOperation.java:67)
at org.activiti.engine.impl.interceptor.CommandInvoker.executeOperation(CommandInvoker.java:73)
at org.activiti.engine.impl.interceptor.CommandInvoker.executeOperations(CommandInvoker.java:57)
at org.activiti.engine.impl.interceptor.CommandInvoker.execute(CommandInvoker.java:42)
at org.activiti.engine.impl.interceptor.TransactionContextInterceptor.execute(TransactionContextInterceptor.java:48)
at org.activiti.engine.impl.interceptor.CommandContextInterceptor.execute(CommandContextInterceptor.java:59)
at org.activiti.spring.SpringTransactionInterceptor$1.doInTransaction(SpringTransactionInterceptor.java:47)
at org.springframework.transaction.support.TransactionTemplate.execute(TransactionTemplate.java:133)
at org.activiti.spring.SpringTransactionInterceptor.execute(SpringTransactionInterceptor.java:45)
```

参数 sql_mode 配置项

去除 `ONLY_FULL_GROUP_BY, NO_AUTO_VALUE_ON_ZERO` 限制，修改方案如下：

查看当前的 sql_mode 参数

如下图存在 `ONLY_FULL_GROUP_BY, NO_AUTO_VALUE_ON_ZERO` 限制，使用 `set global sql_mode` 命令修改，重新登录后生效：

```
percona root@172.16.77.33:core_cloud> select @@sql_mode
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| @@sql_mode                                                                                                                                                            |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| ONLY_FULL_GROUP_BY,NO_AUTO_VALUE_ON_ZERO,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set
Time: 0.008s
percona root@172.16.77.33:core_cloud> set @new_sql_mode := "STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION";
Query OK, 0 rows affected
Time: 0.002s
percona root@172.16.77.33:core_cloud> set global sql_mode = @new_sql_mode
Query OK, 0 rows affected
Time: 0.002s
percona root@172.16.77.33:core_cloud> select @@sql_mode
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| @@sql_mode                                                                                                                                                            |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| ONLY_FULL_GROUP_BY,NO_AUTO_VALUE_ON_ZERO,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set
Time: 0.007s
percona root@172.16.77.33:core_cloud>
Goodbye!
[root@linke-test-H252 .kube]# mycli -h 172.16.77.33 -P3306 -[redacted]
percona 5.7.26-29-log
mycli 1.22.2
Chat: https://gitter.im/dbcli/mycli
Mail: https://groups.google.com/forum/#!forum/mycli-users
Home: http://mycli.net
Thanks to the contributor - spacewander
percona root@172.16.77.33:(none)> select @@sql_mode
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| @@sql_mode                                                                                                                                                            |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set
Time: 0.014s
percona root@172.16.77.33:(none)>
```

使用 set global sql_mode命令修改配置项，重新登录后生效。

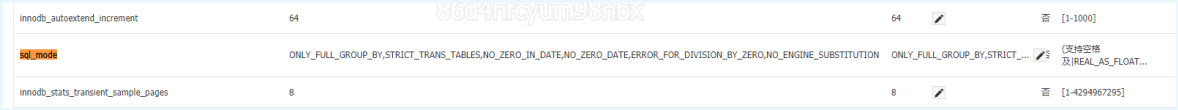
> **说明**
>
> 更多说明请参见：https://codippa.com/how-to-insert-0-in-auto-increment-column-in-mysql/。

无高级账号权限时（如阿里云 RDS，如下图所示），可使用 RDS 控制台修改：

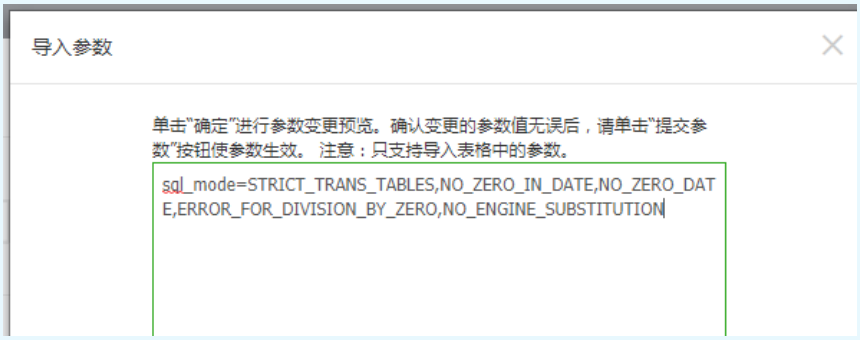
```
MySQL [mysql]> select @new_sql_mode;
+-----------------------------------------------------------------------------------------------------------+
| @new_sql_mode                                                                                             |
+-----------------------------------------------------------------------------------------------------------+
| STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION |
+-----------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)

MySQL [mysql]> SET GLOBAL sql_mode = @new_sql_mode;
ERROR 1227 (42000): Access denied; you need (at least one of) the SUPER privilege(s) for this operation
```

说明

RDS参数修改控制台可能存在 Bug，点击修改时无法找到输入框：



可使用导入参数窗口修改：



影响

不影响部署流程，但影响 antcode 产品的使用、初次登录使用 LinkE 时的模板初始化配置。

参数 lower_case_table_names 配置

在 MySQL 的 `my.cnf` 参数中，将 `lower_case_table_names` 值设置为 `1`，让 MySQL 在执行表查询时对大小写不敏感（所有指定大写的表名在执行期间都会被转换成小写）。

查询 SQL: `show variables where Variable_name='lower_case_table_name` 。

3.4. 创建部署所需服务号

在实际部署前，需要登录 IAM 控制台创建创建 LinkE 部署用服务号。

LinkE 部署涉及两个 AK/SK:

- **管理员登录账号**

  LINKEPORTAL 一次性任务 linkeaccoreinit 使用，以初始化 OP（网关服务）录入 OpenAPI。

> 说明
>
> 推荐使用 antCloudAdmin 预置管理员账号。

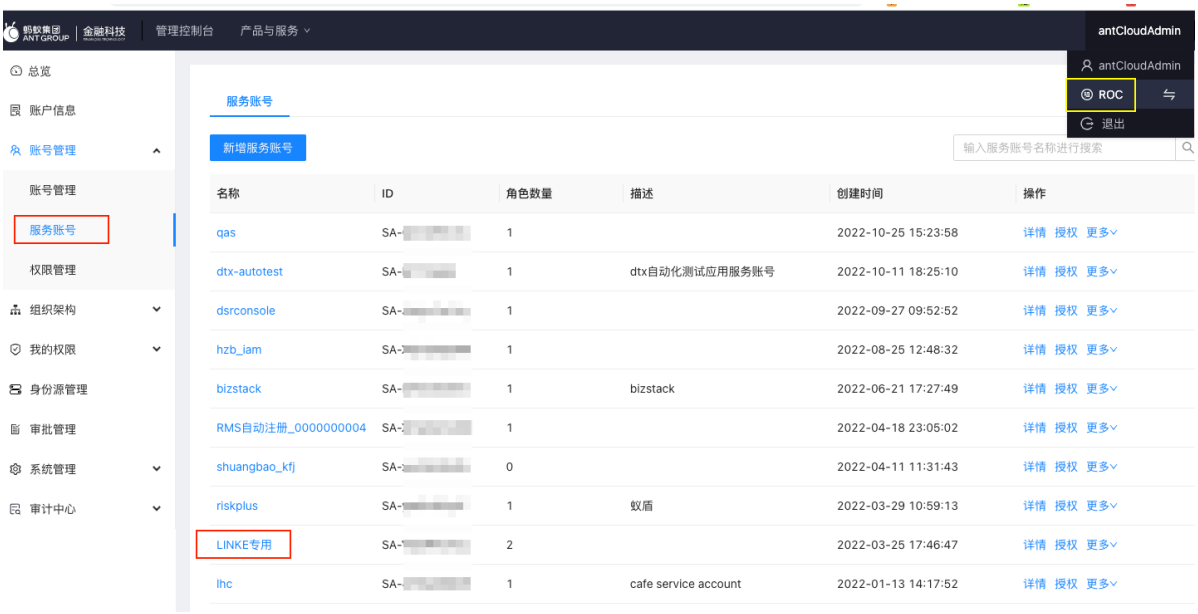
- **ROC 租户管理员服务号**

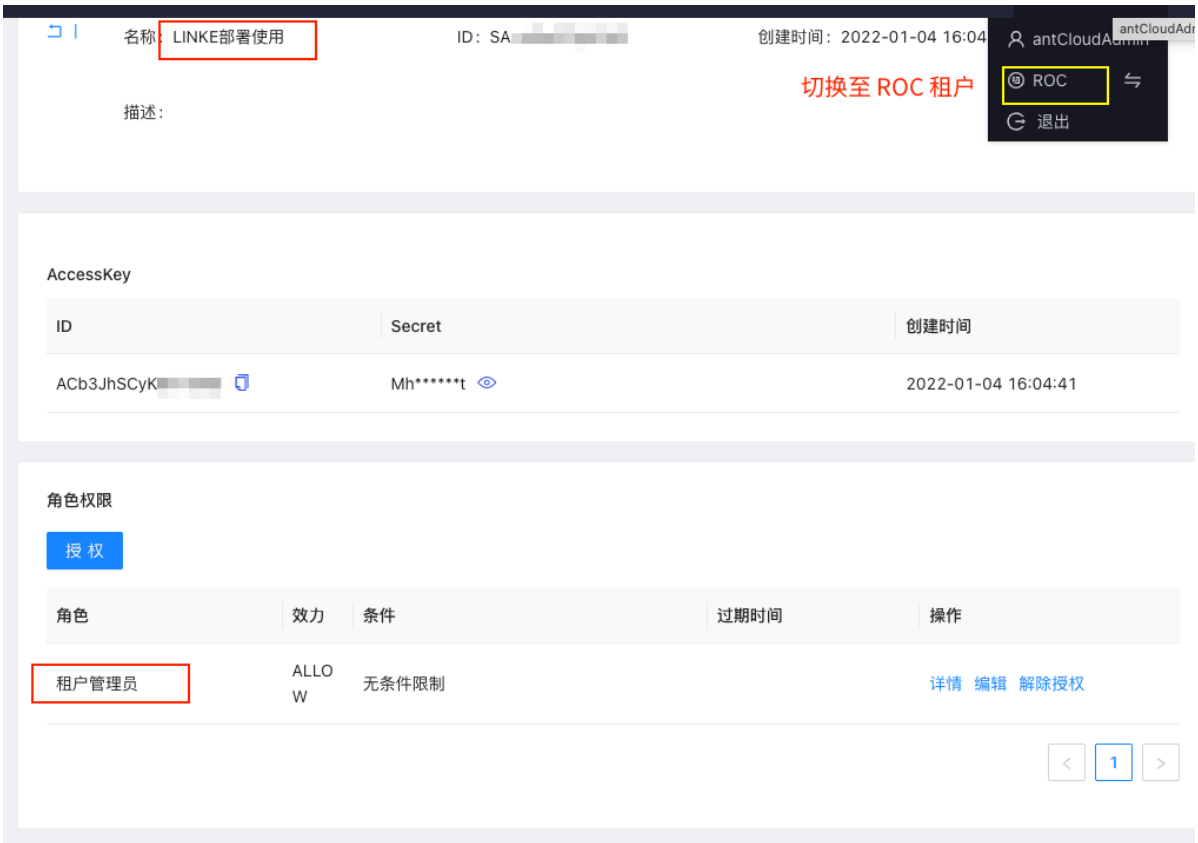
  需要在 IAM ROC 租户上手动创建一个具有租户管理员的账号（名称请带上 LinkE 以标记使用者），然后填写在产品发布公共参数列表：

  - 产品公共参数 LINKEBASE
  - 产品公共参数 LINKEPORTAL

  以下发布任务的参数需要在部署前确认（自动从产品公共参数取数据渲染）：

  - 一次性任务 LINKEBASE linkgwdbinit
  - 应用 LINKEBASE linkucloud
  - 一次性任务 LINKEPORTAL linkeaccoreinit
  - 自动化测试 LINKEPORTAL linkeportaltest


antCloudAdmin
antCloudAdmin
ROC
退出
总览
账户信息
账号管理
账号管理
服务账号
权限管理
组织架构
我的权限
身份源管理
审批管理
系统管理
审计中心
服务账号
新增服务账号
输入服务账号名称进行搜索
名称 ID 角色数量 描述 创建时间 操作
qas SA- 1 2022-10-25 15:23:58 详情 授权 更多
dtx-autotest SA- 1 dtx自动化测试应用服务账号 2022-10-11 18:25:10 详情 授权 更多
dsrconsole SA- 1 2022-09-27 09:52:52 详情 授权 更多
hzb_iam SA- 1 2022-08-25 12:48:32 详情 授权 更多
bizstack SA- 1 bizstack 2022-06-21 17:27:49 详情 授权 更多
RMS自动注册_0000000004 SA- 1 2022-04-18 23:05:02 详情 授权 更多
shuangbao_kfj SA- 0 2022-04-11 11:31:43 详情 授权 更多
riskplus SA- 1 蚁盾 2022-03-29 10:59:13 详情 授权 更多
LINKE专用 SA- 2 2022-03-25 17:46:47 详情 授权 更多
lhc SA- 1 cafe service account 2022-01-13 14:17:52 详情 授权 更多


名称：LINKE部署使用
ID：SA
创建时间：2022-01-04 16:04
描述：
切换至 ROC 租户
antCloudAdmin
ROC
退出
AccessKey
ID Secret 创建时间
ACb3JhSCyK Mh******t 2022-01-04 16:04:41
角色权限
授 权
角色 效力 条件 过期时间 操作
租户管理员 ALLOW 无条件限制 详情 编辑 解除授权
1

4.解决方案

产品集与版本

需要找研发同学确认版本信息。

产品选择

基础产品（必选功能）：LINKEBASE + ANTCODE + LINKB + LINKECI + LINKEPORTAL。

拓扑和规格

- 默认生产部署统一选择 **单机房标准部署 > 标准生产-单机房**。
- POC 部署统一选择 **单机房标准部署 > 标准 POC**。

解决方案制作

如果部署 Antstack Plus 环境，请确保已经为环境设置打标，如下图所示。

5.后端应用部署

5.1. LINKEBASE

5.1.1. 部署说明

按云游默认顺序进行部署，其中 linkemongo 应用需要设置部署分组策略，调整为一台一台分组部署（就是 beta分组，默认策略是直接先部署两台，再部署一台）。

无需关注自动化测试应用的运行状态，可跳过，LinkE 产品的自动化测试应用统一配置在 LINKEPORTAL 产品中。

> 说明
>
> - 在部署 LINKEBASE 前，需要确定环境中已完成外部依赖产品的部署。
> - 产品公共参数中，需填入 IAM ROC 租户管理员服务号的 AK/SK，详见前文 创建部署所需服务号。

5.1.2. 部署方式

云游导入的 LinkE 解决方案中直接单击发布部署，首次部署可以只分一组部署。

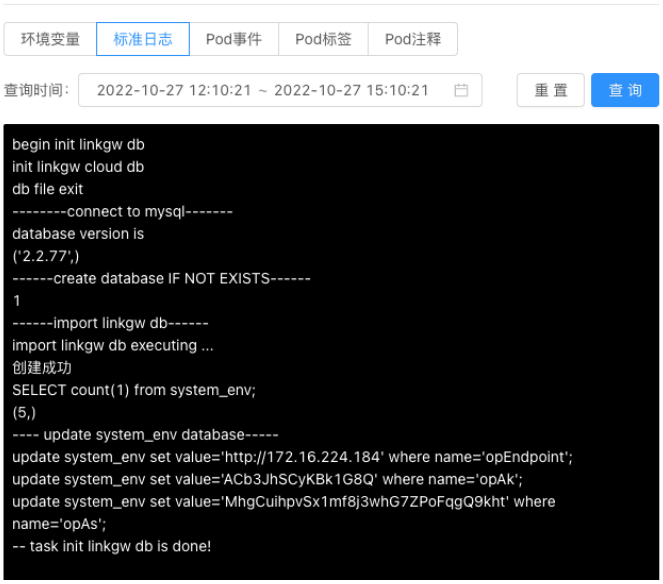
应用初始化

> 说明
>
> - 在进行应用初始化时，一次性任务（job）initlinkebasedb 和 linkgwdbinit 的必须运行成功。
> - linkemongo 与 linkuredis 为开源应用，非常规 SOFA 技术栈 Java 程序应用，部署流程较为特殊。

linkgwdbinit

需要进入数据库 linkgw_cloud 检查是否成功写入相关表。

- **云游可查看 Pod 日志**



- **进入数据表确认**

  需要确认 linkgw_cloud 库存在如下表。

> 说明
>
> 不能是空库。

linkemongo

LINKEBASE 中的 linkemongo 为有状态应用，需要持久化数据（且数据非常重要），当前以 PVC 方式挂载在 Pod 的 `/data` 目录下，用以存储 mongodb 数据。

部署常见问题

第一次部署 linkemongo 无法启动，多见于卸载后重新部署时复用上一次的持久化存储，健康检查一直不通过。

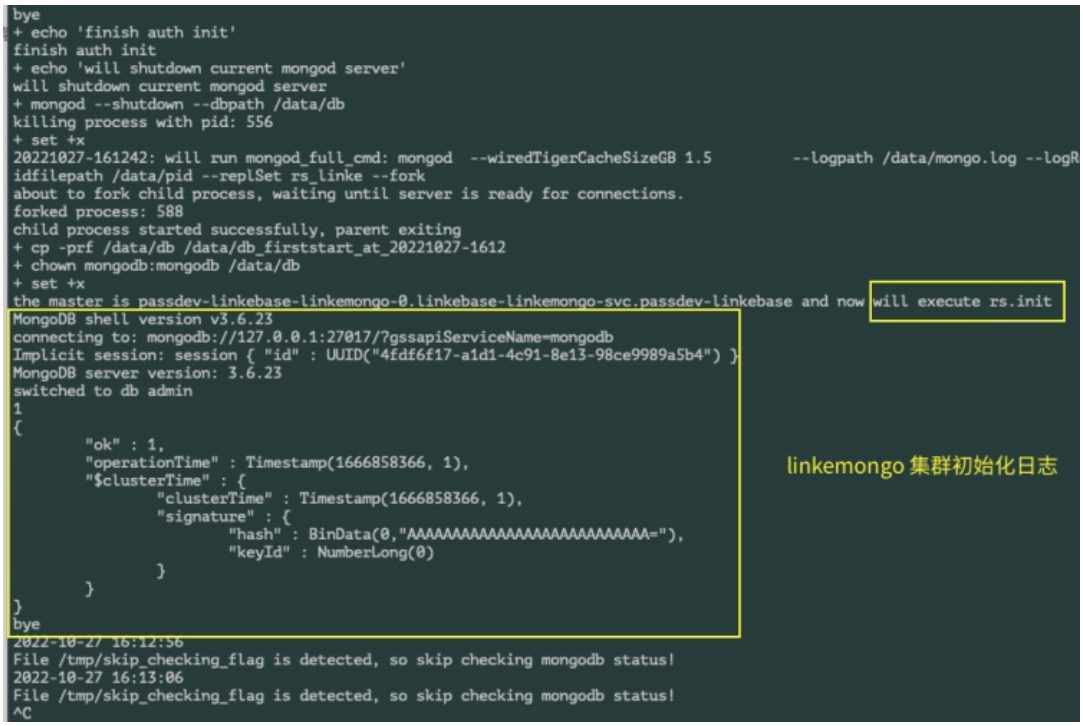
解决方案

> 说明
>
> 以下解决方案仅适用于第一次部署，而且仅适用 mongo0 号节点直接没启动成功，mongo1 号节点还没有开始部署的场景，须明确！

使用云游 webshell 进入 Pod：

```
# 1. 创建空文件，/opt/start-local-mongodb.sh 脚本有埋点
touch /tmp/skip_checking_flag

# 2. 备份当前 /data 目录
cd /data
mkdir backup
mv * backup

# 3. kill mongod，rinetd 进程
ps -ef | grep mongod
kill $mongod_pid
ps -ef | grep rinetd
kill $rinetd_pid

# 4. 重新初始化 mongodb 启动程序
bash /opt/start-local-mongodb.sh

# 5. 查看日志，待 mongodb 初始化成功后（见截图），ctrl-c 终止此 shell 脚本
# ...

# 6. 移除标志位文件
mv /tmp/skip_checking_flag /tmp/skip_checking_flag2
```

集群初始化日志截图，然后重新尝试健康检查 linkemongo pod 状态。



其他情况说明

若 mongo1 号节点启动后，mongo0 号节点又报健康检查不通过，说明 mongo1 的 `/data` 目录上次卸载时没清除干净，而且也运行着 mongod 程序，这时两个节点数据不一致集群无法建立，所以需要先清空 mongo1 节点 `/data` 目录，停止 mongod 进程后，再 `touch /tmp/skip_checking_flag` 文件后，执行重发操作。

数据备份

> 重要
>
> 容器目录 `/data/db` 是 linkemongo 容器的运行时数据，非常重要，需每天定时备份，站点运维人员应知晓。

- 阿里云底座应直接在 IaaS 层创建磁盘自动快照策略，每天自动备份。
- 在 Pod 中也添加定时任务（下图所示 `daily_operate_mongo.py` 脚本），每天凌晨备份 mongodb 全量数据至 `/data/mongodump` 文件夹。

```
root@asa-linkebase-linkemongo-0:/# crontab -l
21 5 * * * python3 /opt/daily_operate_mongo.py >> /tmp/daily_operate_mongo.log 2>&1
01 01 * * * bash /opt/clean-data.sh >> /tmp/clean-data.log 2>&1
```

> 说明
>
> - 备份条件：磁盘有充足的剩余空间。
> - 备份清除策略：保留最近 15 天的数据，具体见脚本。

linkuredis

LINKEBASE 中的 linkeredis 同样为有状态应用，需要持久化数据，但数据并不重要（缓存数据）。

当前也以 PVC 方式挂载在 Pod 的 `/data` 目录下，Redis 的内存自动快照数据存储于此。

端口检查

与开源默认解决方案不同（检查 rinetd 程序，其将 SLB 流量导入集群主节点）说明如下：

- linkemongo 检查 27018 端口（rinetd 端口转发程序），非 mongodb 程序默认的 27017 商品。
- linkuredis 检查 6378 端口（rinetd 端口转发程序），非 redis 程序默认的 6379 商品。

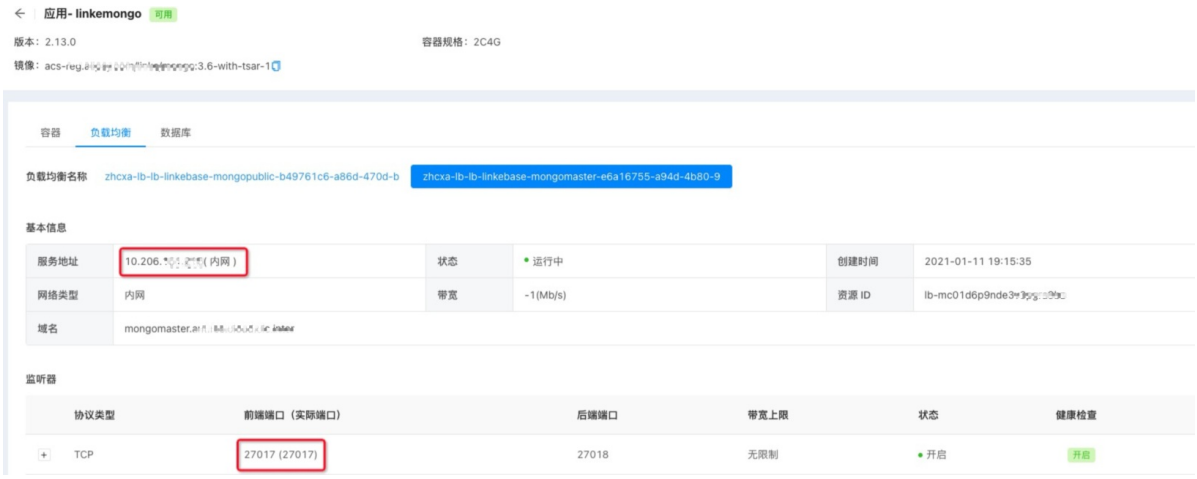
5.1.3. 部署检查

linkemongo 检查

- **MongoDB 集群状态检查**

  查询 mongo 的 SLB 地址，登录到一个 mongo 的容器，执行如下命令。

```
mongo
# 不需要指定其他参数，使用默认端口 27017

# 登录后再输入
use admin
db.auth("linke", "LinkE2017")

// 预期显示1，代表 TRUE，鉴权通过
// 再输入

rs.status()
// 预期展示集群状态
```




```
rs_linke:PRIMARY> use admin;
switched to db admin
rs_linke:PRIMARY> db.auth('linke','LinkE2017');
1
rs_linke:PRIMARY> rs.status()
{
        "set" : "rs_linke",
        "date" : ISODate("2021-03-02T10:11:10.122Z"),
        "myState" : 1,
        "term" : NumberLong(4),
        "heartbeatIntervalMillis" : NumberLong(2000),
        "optimes" : {
                "lastCommittedOpTime" : {
                        "ts" : Timestamp(1614679870, 2),
                        "t" : NumberLong(4)
                },
                "readConcernMajorityOpTime" : {
                        "ts" : Timestamp(1614679870, 2),
                        "t" : NumberLong(4)
                },
                "appliedOpTime" : {
                        "ts" : Timestamp(1614679870, 2),
                        "t" : NumberLong(4)
                },
                "durableOpTime" : {
                        "ts" : Timestamp(1614679870, 2),
                        "t" : NumberLong(4)
                }
        },
        "members" : [
                {
                        "_id" : 0,
                        "name" : "10.206.
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 2513249,
                        "optime" : {
                                "ts" : Timestamp(1614679864, 1),
                                "t" : NumberLong(4)
                        },
                        "optimeDurable" : {
                                "ts" : Timestamp(1614679864, 1),
                                "t" : NumberLong(4)
                        },
                        "optimeDate" : ISODate("2021-03-02T10:11:04Z"),
                        "optimeDurableDate" : ISODate("2021-03-02T10:11:04Z"),
                        "lastHeartbeat" : ISODate("2021-03-02T10:11:08.271Z"),
                        "lastHeartbeatRecv" : ISODate("2021-03-02T10:11:08.258Z"),
                        "pingMs" : NumberLong(0),
                        "syncingTo" : "10.206.
                        "configVersion" : 1
```

> **说明**
>
> 在 MongoDB 的三个节点都尝试运行一遍，只有出现三个节点（members 中一个 primary 两个 secondary）且 state 均正常时才算部署正常。

- **MongoDB 主节点端口转发检查**

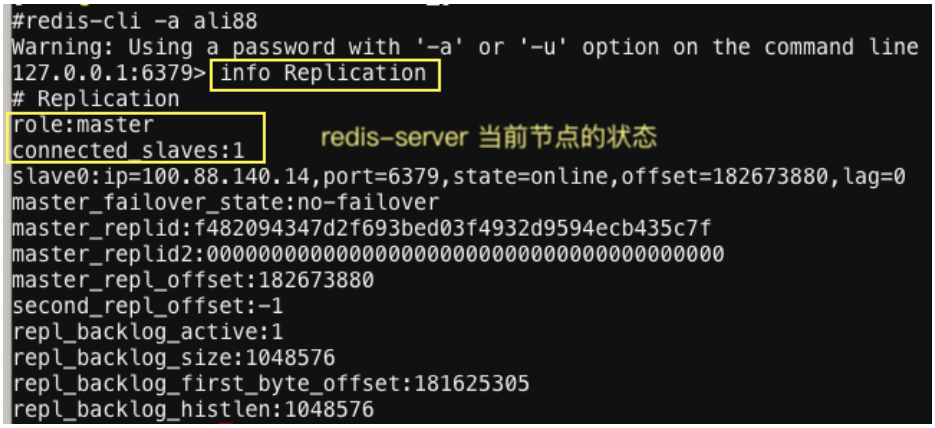
选择一台 secondary 节点，输入 `mongo --port 27018`，预期进入 mongoprimary 节点。

原因：mongodb 以副本集模式安装，仅主节点可接入写入请求，因此程序使用 27018 端口使用 SLB 地址连接 mongodb，此端口的流量被转发到 primary 节点的 27017 端口上。

linkuredis 检查

进入 linkuredis 的每个节点，输入 `redis-cli -p 6378 -a ali88` 后都可以连接到 redis 服务的主节点。

在 redis-cli 控制台上输入 `info Replication` 可以看到主从状态，同 redis pod 副本数一致（默认 1 主 1 从）。



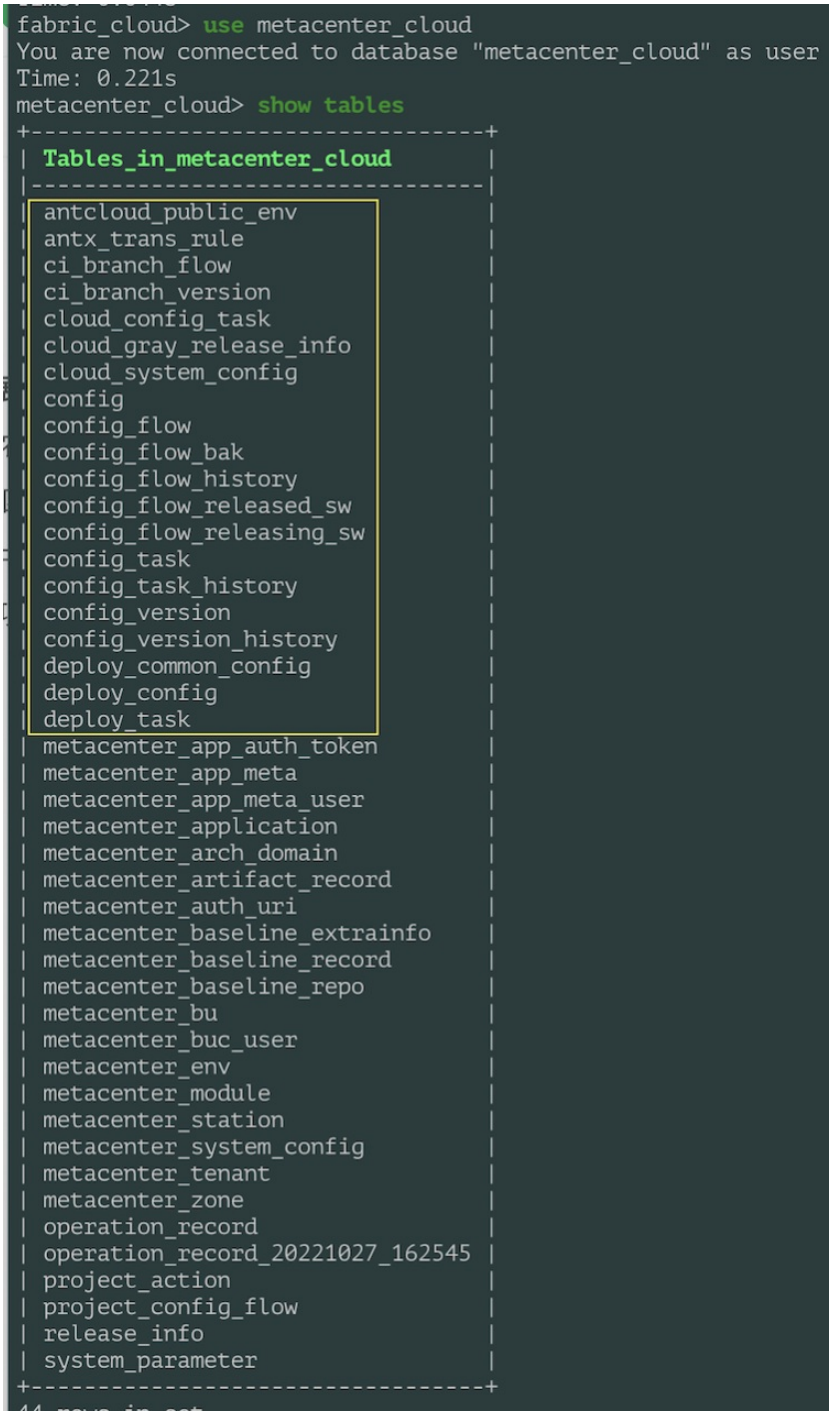
守夜人巡检

以上 linkemongo 与 linkuredis 应用的端口检查项已接入守夜人巡检 HAS 平台，站点运维人员可自助导入。

LinkM 数据库检查

应用 LinkM 在启动前，会将 fabric_cloud 中的数据表迁移至本库供 LinkM 程序访问，因此 LinkM 程序启动后需要确认 metacenter_cloud 库中存在以下数据表。

> **说明**
>
> 数据表源自一个已下载的应用 fabriccloud，迁移程序写在 LinkM 应用的启动脚本中。



LinkU 数据库检查

第一次部署后，应查看 linku_cloud 数据库的 linku_secret_config 表是否与 IAM ROC 租户下 LINKE 使用的租户管理员服务号 AK/SK 一致，endpoint 是否为 OP 产品的 SLB IP 地址。

> **说明**
>
> 若不一致需要需要在产品公共参数中录对，再重发 LinkU 应用。

SQL 为：`select * from linku_secret_config where `type` = "CLOUD_ACCESS" \G` 。

```
mysql root@11.165.57.69:linku_cloud> select * from linku_secret_config
  where `type` = "CLOUD_ACCESS" \G;
***************************[ 1. row ]***************************
id           | 41
gmt_create   | 2022-03-16 17:52:11
gmt_modified | 2022-03-16 17:52:11
type         | CLOUD_ACCESS
name         | linku-system:defaultCloudAccessConfig
value        | {
"accessKeyId":"ACb3JhSCyKBk1G8Q",
"accessKeySecret":"[redacted]",
"endpoint":"http://172.16.224.184/gateway.do"
}
unique_key   | defaultCloudAccessConfig
namespace    | linku-system
deleted      | 0
description  | <null>

1 row in set
Time: 0.024s
```

OP 产品的 SLB IP 地址位置如下图所示：

5.2. ANTCODE

5.2.1. 部署说明

> 说明
>
> - 在部署 ANTCODE 前，需要完成 LINKEBASE 的部署。
> - 按云游默认顺序进行部署，其中 antcodenode 应用需要设置部署分组策略，调整为一台一台分组部署（就是 beta 分组）。
> - 无需关注自动化测试应用的运行状态，可跳过，LinkE 产品的自动化测试应用统一配置在 LINKEPORTAL 产品中。

应用配置规划

ANTCODE 中的 antcodenode 为有状态应用，需要持久化数据（且数据非常重要），当前以 PVC 方式挂载在 Pod 的 `/home/git` 目录下，用以存储客户的代码数据。

数据备份

> 重要

容器 antcodenode 目录 `/home/git/repo` 是客户代码的生产数据，非常重要，需每天定时备份，站点运维人员应知晓。

- 阿里云底座应直接在 IaaS 层创建磁盘自动快照策略，每天自动备份。
- 物理机底座需依据物理机器本身的数据备份策略，需要站点运维人员通过守夜人平台进行每天自动备份。

> 说明
>
> 如因代码数据过大，无法在宿主机中执行定时任务备份，可备份到外部的对象存储服务中。

5.2.2. 部署方式

云游导入的 LinkE 解决方案中直接单击发布部署，首次部署可以只分一组部署。

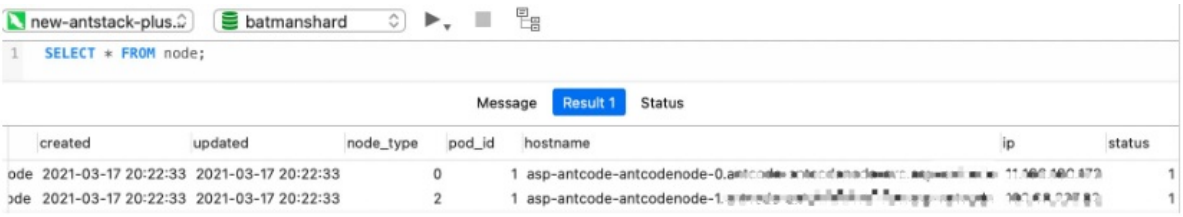
> 说明
>
> 在进行应用初始化时，一次性任务（job）antcodedatainit、codecenterinit 必须运行成功。

5.2.3. 部署检查

antcodenode 检查

查询 batmanshard 的数据库中的 node 表数据，IP 字段值需要为 IP 地址信息，和 antcodenode 容器 IP 对应。

```
select * from node;
```



部署完成后有两条 status=1 的记录：

- 一条对应容器 `antcodenode-0 node_type`，数值为 0 。
- 一条对应容器 `antcodenode-1 node_type`，数值为 2。

5.3. LINKB

5.3.1. 部署说明

> 说明
>
> 在部署 LINKEB 前，需要完成 LINKEBASE 和 ANTCODE 的部署。

按云游默认顺序进行部署，无需关注自动化测试应用的运行状态，可跳过，LinkE 产品的自动化测试应用统一配置在 LINKEPORTAL 产品中。

> 重要

LinkE 的构建服务要基于底座的能力进行镜像的构建，其部署依赖宿主机节点的服务和配置，请务必按照前文 确认宿主机内核参数配置 中的操作调整宿主机的内核参数配置。

buildrunner 应用需要使用宿主机的内核虚拟化能力在容器中进行镜像构建。

```
echo "user.max_user_namespaces=28633" >> /etc/sysctl.conf &&  sysctl -p
```

5.3.2. 部署检查

云游 webshell 进入 buildrunner 容器，输入 `supervisorctl` 进入托管进程控制台：

```
sh-4.2# bash
[root@asa-linkb-buildrunner-0 ~]# supervisorctl
buildkitd-rootless               RUNNING   pid 25, uptime 15 days, 6:36:34
checking_build_tasks             RUNNING   pid 24, uptime 15 days, 6:36:34
flash                            RUNNING   pid 26, uptime 15 days, 6:36:34
supervisor>
```

预期三个进程都是正常的 RUNNING 状态，运行持续时间也保持一致。

部署常见问题

- **没有调整宿主机的内核参数配置，导致 supervisorctl 控制台中 buildkitd 进程无法启动。**

  解决方案：找到 buildrunner 所在宿主机，重新配置内核参数后，使用云游重启对应 buildrunner 容器。

- **集群中只有部分宿主机支持此内核参数配置。**

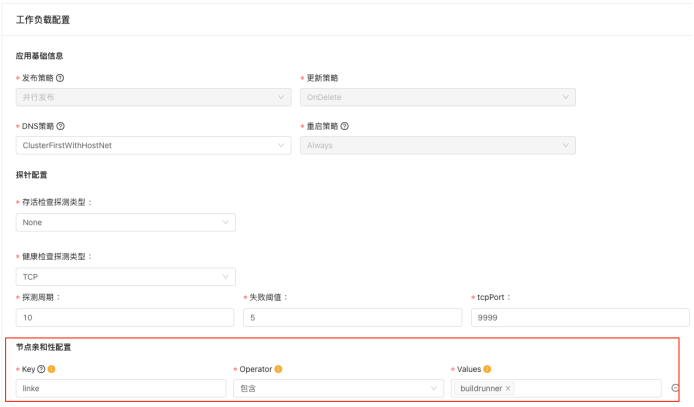
  解决方案：

  i. 先在云游 local 解决方案中为 buildrunner 容器修改应用调度配置，增加 node selector 标签（如标签 `linke:buildrunner` ）。

  ii. 在 captain / ack 中为指定节点添加同样的标签，限制 buildrunner 容器只能被调度到指定节点上。

  操作示例：

  i. 云游 local 应用配置的节点亲和性配置示例。



  ii. Captain 修改节点标签示例。

- **集群中所有宿主机都不支持此内核参数配置。**

  解决方案：

  - 首选方案是升级 Linux 操作系统或内核版本，最新版支持此参数。
  - 次选方案是改用 Docker 构建方案（Docker In Docker 构建方案，需要在宿主机上启动 dockerd 进程，不推荐）。

5.4. LINKECI

5.4.1. 部署说明

按云游默认顺序进行部署，无需关注自动化测试应用的运行状态，可跳过，LinkE 产品的自动化测试应用统一配置在 LINKEPORTAL 产品中。

> 说明
>
> 在部署 LINKECI 前，需要完成 LINKEBASE、ANTCODE 和 LINKB 的部署。

需要确认一次性任务 aciinitscript 如下部署参数符合预期：

- ACLINKELIB_BUILD_AKS_FRAMEWORK：**dockerBuild-v3**。

- ACLINKELIB_BUILD_AKS_FRAMEWORK_VERSION：**1.0.0**。

常量	ACLINKELIB_BUILD_AKS_FRAMEWORK 非空	dockerBuild-v3	aks build framework name
常量	ACLINKELIB_BUILD_AKS_FRAMEWORK_VERSION 非空	1.0.0	aks build framework version
常量	ACLINKELIB_IMAGE_COPY_FRAMEWORK 非空	ImageTag	image copy framework name
常量	ACLINKELIB_IMAGE_COPY_FRAMEWORK_VERSION 非空	v1.0.0	image copy framework version

5.4.2. 部署方式

云游导入的 LinkE 解决方案中直接单击发布部署，首次部署可以只分一组部署。

> 说明
>
> 在进行应用初始化时，一次性任务（job）aciinitscript、linkainitscript、jenkinsregister 必须运行成功。

5.4.3. 部署检查

1. 云游容器列表搜索 aci。

   找到 acicloud, acijenkinscloud 两个容器，可能存在多个实例：

2. 验证 acijenkinscloud 每个实例可在浏览器中使用 `ip:8080` 访问。

   如上图例，acijenkinscloud 的实例 IP 地址是 11.158.125.42，则输入 `http://11.158.125.42:8080`，可访问主页并能使用 admin / ciAdmin 登录，如下图所示：

3. 验证 acicloud 实例已录入 acijenkinscloud 的实例信息。

   如上图所示，acicloud 实例 IP 地址是 11.158.125.44，则输入 `http://11.158.125.44/index.html` 。

   > 说明
   >
   > 若存在多实例可使用任一 IP 地址访问。

   如下图所示，其已自动录入 acijenkinscloud 所有实例 IP 地址列表，并且实例状态为在线，实例标签为 `v2` 。

5.5. LINKEPORTAL

5.5.1. 部署说明

> 说明
>
> - 在部署 LINKEPORTAL 前，需要完成 LINKEBASE、ANTCODE、LINKB、LINKECI 的部署。
> - 产品公共参数中，需填入 IAM ROC 租户管理员服务号的 AK/SK，详见 创建部署所需服务号。

5.5.2. 部署方式

云游导入的 LinkE 解决方案中直接单击发布部署，首次部署可以只分一组部署。

> 说明
>
> 在进行应用初始化时，一次性任务（job）linkeportalinit 的一次性任务必须运行成功。

注意事项

- linkeaccoreinit 一次性任务部署

- 需要确认与 IAM 与 OP 交互的任务，详见下文 一次性任务 linkeaccoreinit 发布方法。
- 发布前应确认 IAM 与 OP 应用（主要是 aciamcore，acprodapigw）是否运行正常。

- linkeport alinit 一次性任务部署

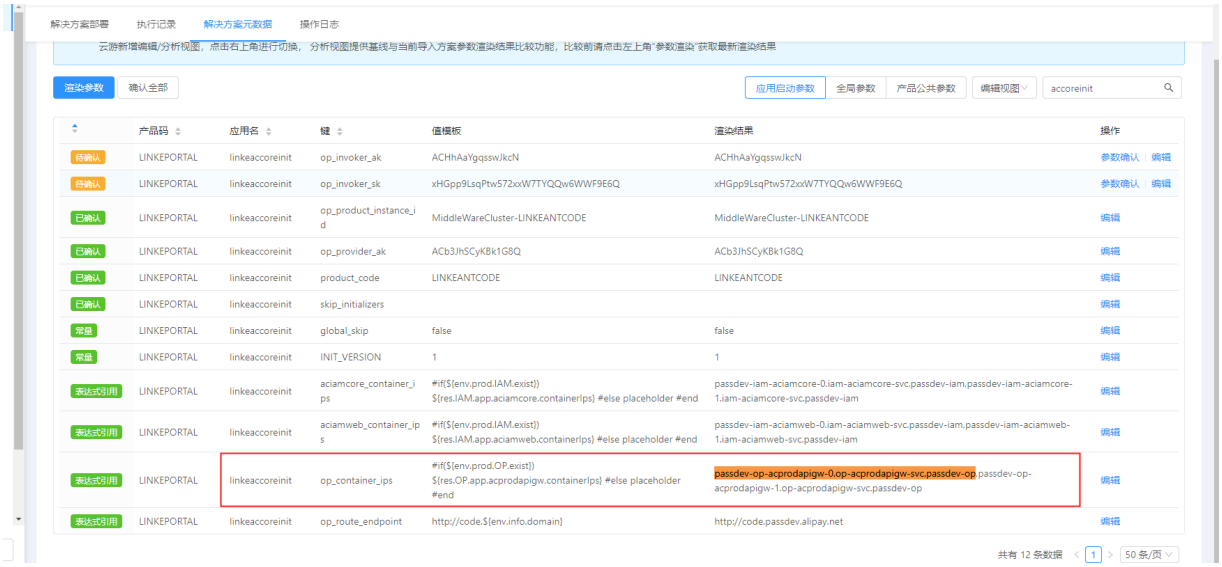
  产品公共参数中，需填入 IAM ROC 租户管理员服务号的 AK/SK，详见 创建部署所需服务号。

> **说明**
>
> 若没有填写正确，则会造成在 IAM 上已登录，却一直提示未登录不停跳转至登录页的异常现象。

一次性任务 linkeaccoreinit 发布方法

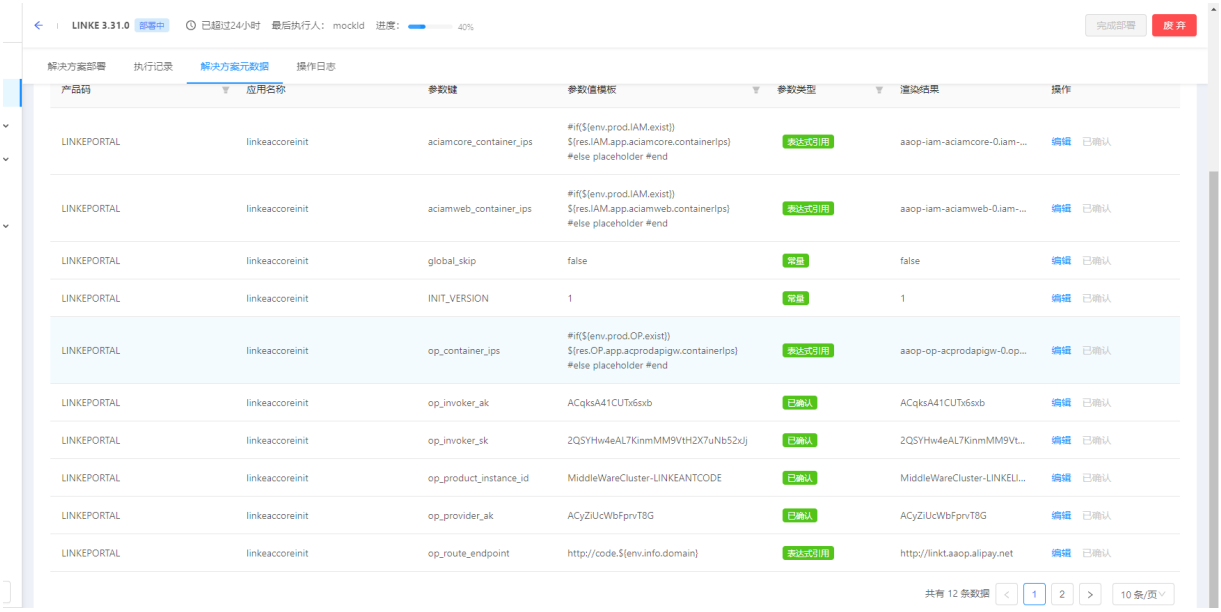
发布前置检查

当前 linkeaccoreinit 发布参数中指定多个 acprodapigw 应用实例地址，但 OP 应用的代码中写死了 ips[0] 作为 endpoint，因此若编号为 1 的 Pod 存活但编号为 0 挂掉，OP 侧无法录入 openapi，需要编号为 0 的 Pod 存活才能运行成功。



如下图所示，acprodapigw-0 应用实例下线就无法发布 linkeaccoreinit。

参数截图



需要现场填写的参数

> **重要**
>
> `op_invoker_ak` 、 `op_invoker_sk` 、 `op_provider_ak` 参数如果第一次没有填对，后续修改后再次去执行一次性任务，由于 OP 有缓存很可能导致参数无法修改成功，所以尽可能第一次填对。

参数名	参数值	描述
product_code	需要多次填写并重新发布 • LINKEANTCODE • LINKE • LINKEDEPLOYCORE 如第一次填写 LINKEANTCODE 并发布，第二次改成 LINKE 并发布，第三次改成 LINKEDEPLOYCORE 再发布。	对应 OPM 上的产品码，导入多个应用元数据用逗号隔开，OP 初始化任务一次仅只能初始化一个产品码，故需要发多次。LinkE 应用共对应如下几个产品码，但只需要发左列那几个：LINKE、LINKELINKFLOW、LINKEDEPLOYCORE、LINKEANTCODE、LINKEFABRIC。

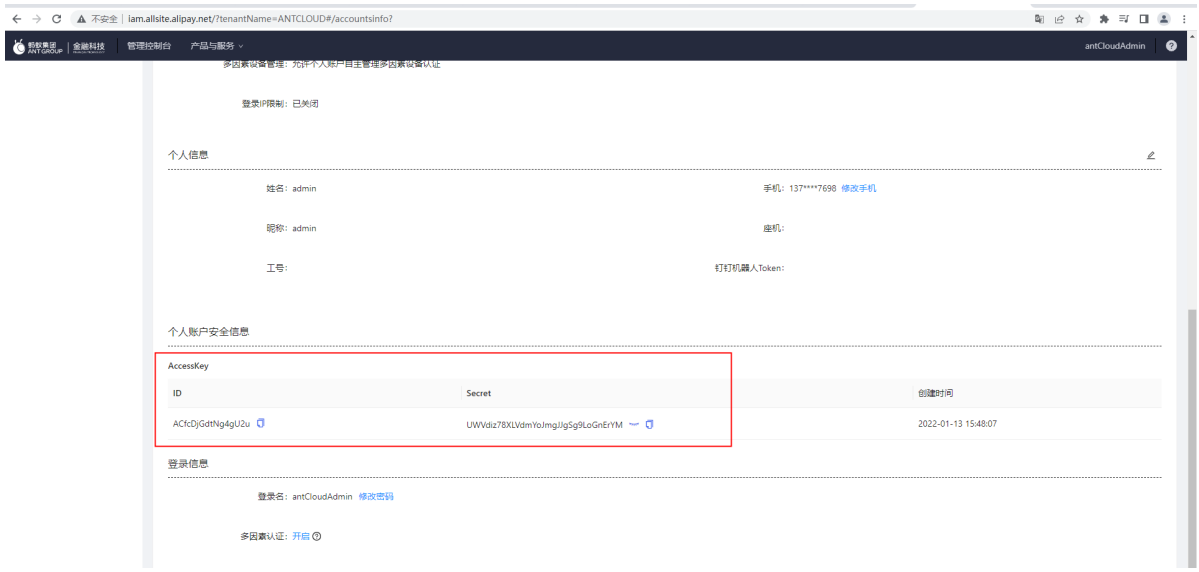
op_product_instance_id	需要多次填写并重新发布 MiddleWareCluster-LINKEANT CODE，MiddleWareCluster-LINKE，MiddleWareCluster-LINKELINKT。	网关集群路由的集群 id（现场填写，需填写三次） 部署时需要对应 product_code 参数依次更改为：MiddleWareCluster-LINKEANT CODE，MiddleWareCluster-LINKE，MiddleWareCluster-LINKELINKT 触发初始化工作。
op_route_endpoint	需要多次填写并重新发布 http://code.${env.info.domain}， http://linke.${env.info.domain}， http://linkt.${env.info.domain}。	网关路由的下游地址（现场填写，需填写三次），对应 op_product_instance_id 参数填写的内容依次为： http://code.${env.info.domain}， http://linke.${env.info.domain}， http://linkt.${env.info.domain}
op_invoker_ak	在 IAM 控制台上获取	用户 AK，下游调用 IAM 做鉴权时，所用的AK，最好使用管理员用户 AK，不然会有权限问题（现场填写，只填一次即可）。
op_invoker_sk	在 IAM 控制台上获取	用户 SK，下游调用 IAM 做鉴权时所用的 SK（现场填写，只填一次即可）。
op_provider_ak	在 IAM 控制台上获取	网关后端验签所用 AK，此处是 POC 租户下 LINKE 专用 AK（现场填写，只填一次即可）。

AK/SK获取方法

详情请参见 LinkE 部署涉及两个 AK/SK。

- **op_invoker_ak和op_invoker_sk 获取方法**

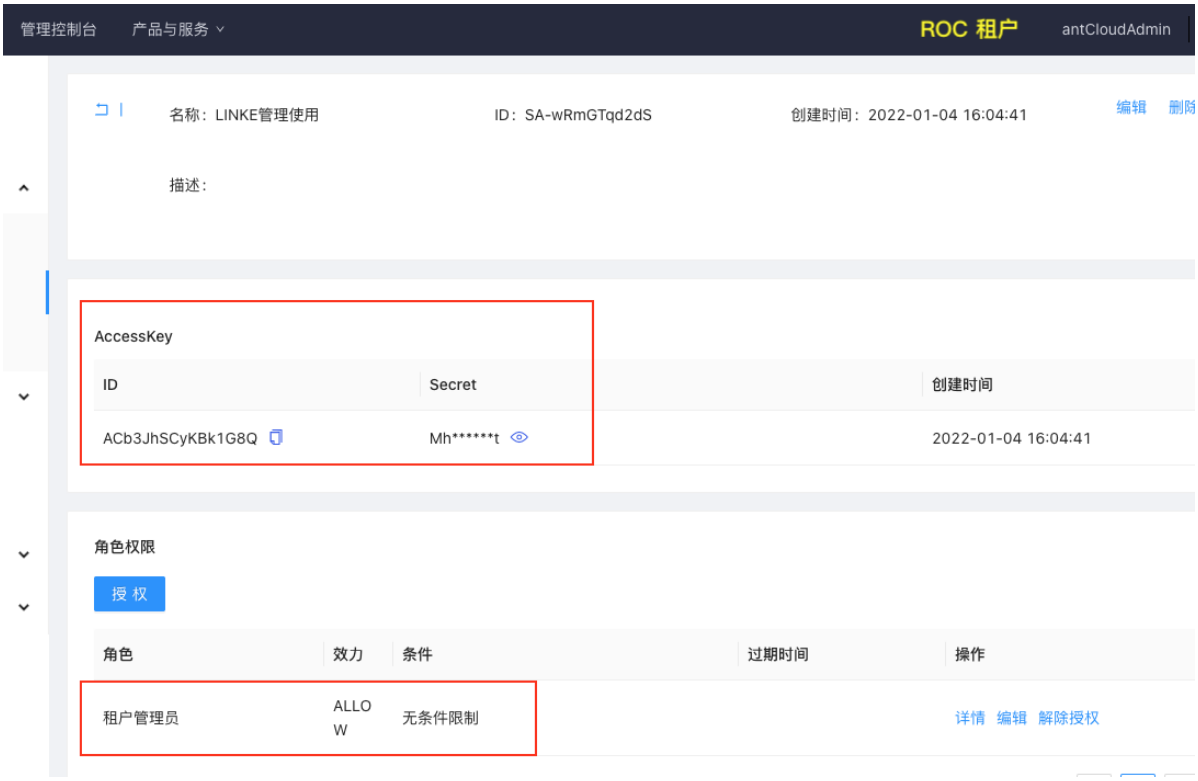
  使用管理员账号登录 IAM 上可看到该账号的 AK、SK，`op_invoker_ak` 对应 AccessKey，`op_invoker_sk` 对应 Secret。



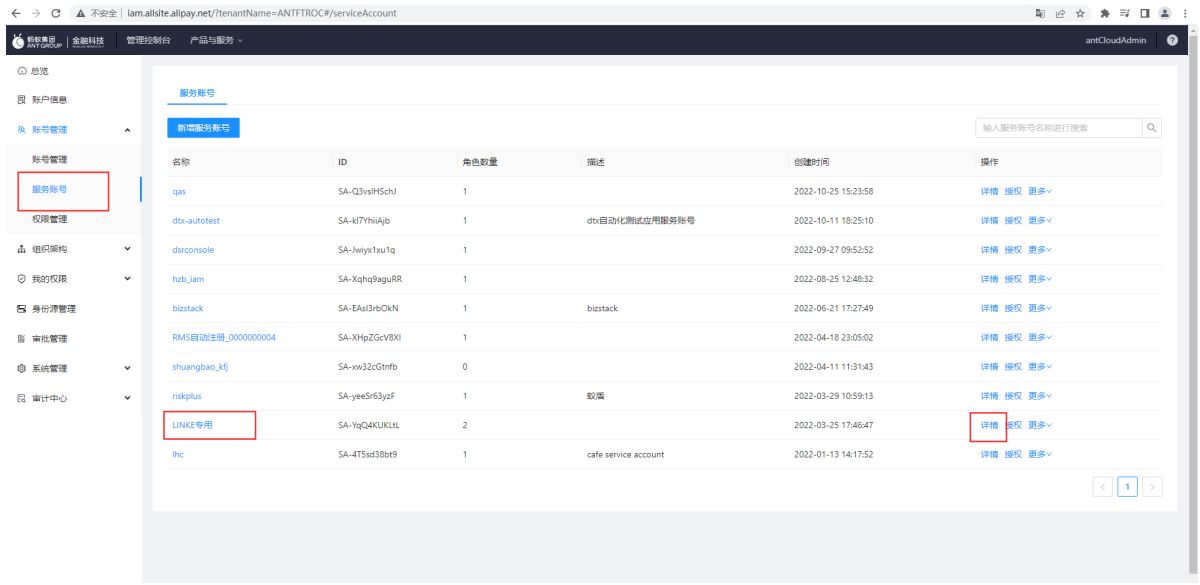
- **op_provider_ak 获取方法**

  切换至 ROC 租户下。

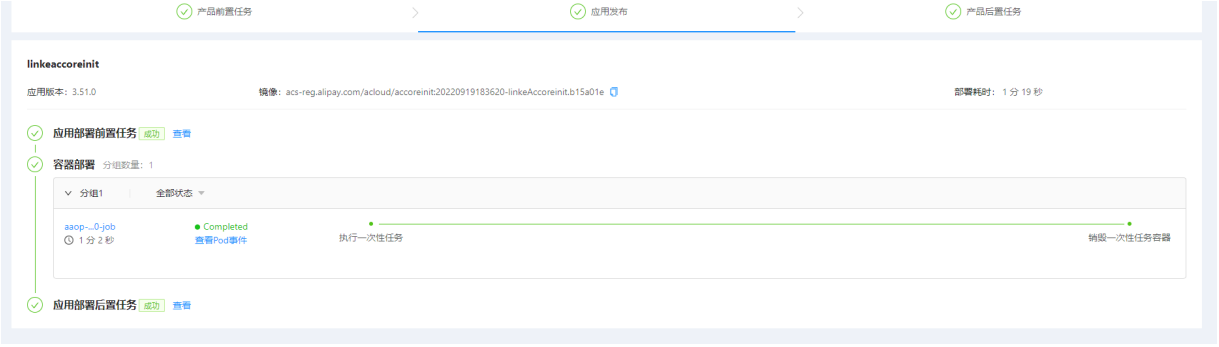
登录具有租户管理员权限的服务号，详情中有 AccessKey，就是 op_provider_ak 的值。



“**LINKE专用**”服务账号默认有此权限，可复用。

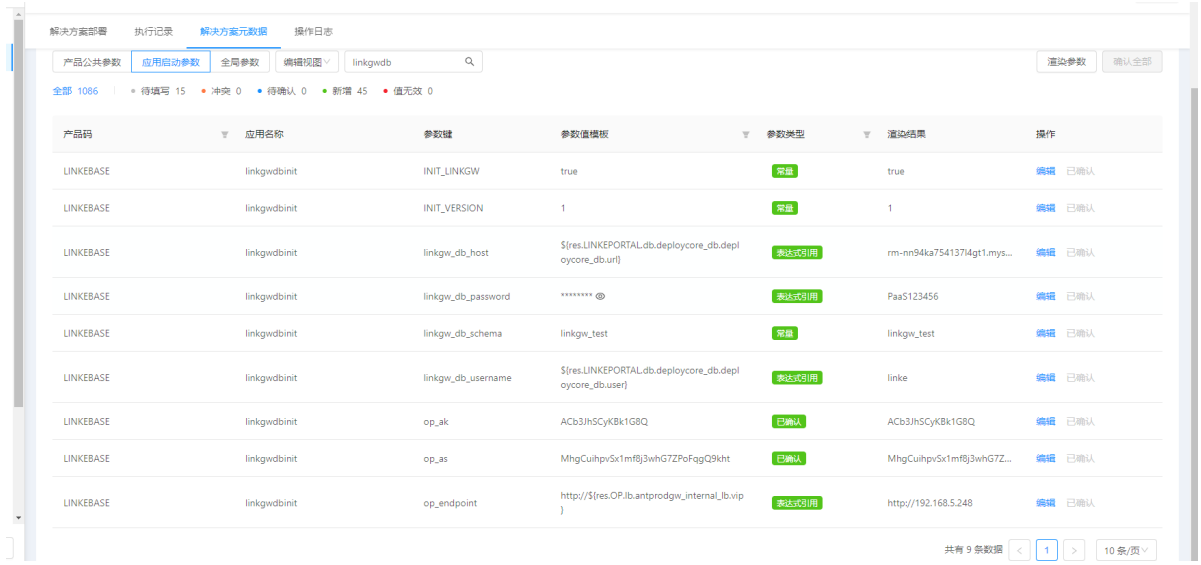


云游pod运行结果



linkgwdbinit 发版方案

- **参数截图**



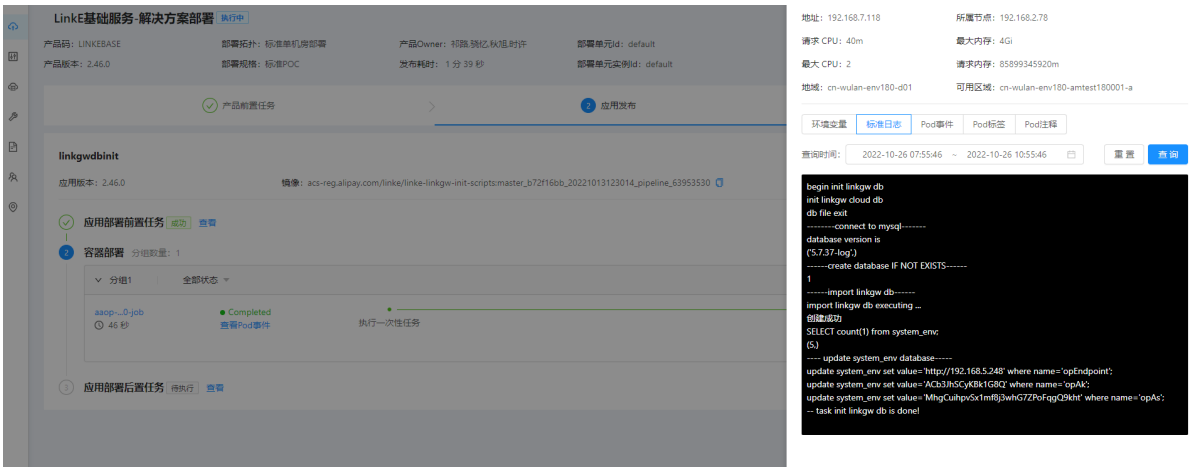
- **需要现场填写的参数**

参数名	参数值	描述

op_ak	在 IAM 控制台上获取	网关后端验签所用 AK，此处是 POC 租户下 LINKE 专用 AK（现场填写）。
op_as	在 IAM 控制台上获取	网关后端验签所用 AS，此处是 POC 租户下 LINKE 专用 AS（现场填写）。

> 说明
>
> op_ak 和 op_as 的获取方式可参照前文 op_provider_ak 的获取方式。

- **云游 Pod 运行结果**



5.5.3. 部署检查

第一次部署后，应查看 linku_cloud 数据库的 linku_secret_config 表是否与 IAM ROC 租户下“LINKE专用”服务号的 AK/SK 一致。若不一致需要查看常见问题解决方案：LinkE 产品完成部署后账号无法登录。

```
mysql root@11.165.57.69:linku_cloud> select * from linku_secret_config
  where `type` = "CLOUD_ACCESS" \G;
***************************[ 1. row ]***************************
id           | 41
gmt_create   | 2022-03-16 17:52:11
gmt_modified | 2022-03-16 17:52:11
type         | CLOUD_ACCESS
name         | linku-system:defaultCloudAccessConfig
value        | {
"accessKeyId":"ACb3JhSCyKBk1G8Q",
"accessKeySecret":"[redacted]",
"endpoint":"http://172.16.224.184/gateway.do"
}
unique_key   | defaultCloudAccessConfig
namespace    | linku-system
deleted      | 0
description  | <null>

1 row in set
Time: 0.024s
```

6.前端页面部署

6.1. 客户访问域名配置

操作步骤

在用户的本地电脑配置后访问，或者在客户办公网的 DNS 服务器上注册域名。

- 2211 及以后版本域名配置如下：

```
# 必须的域名配置
linkconsole.${env.info.domain}  #解析到 linkeonex（负载均衡）或 bahamut 容器
linkeapi.${env.info.domain}  #解析到 linkeonex（负载均衡）或 minionex 容器
code.${env.info.domain}  #解析到 antcode_slb 或 antcodeweb 容器
```

- 2209 版本域名配置如下：

```
# 必须的域名配置
linkconsole.${env.info.domain}  #解析到 oneconsole（负载均衡）或 oneconsole 容器
linkeapi.${env.info.domain}  #解析到 minionex（负载均衡）或 minionex 容器
code.${env.info.domain}  #解析到 antcode_slb 或 antcodeweb 容器
```

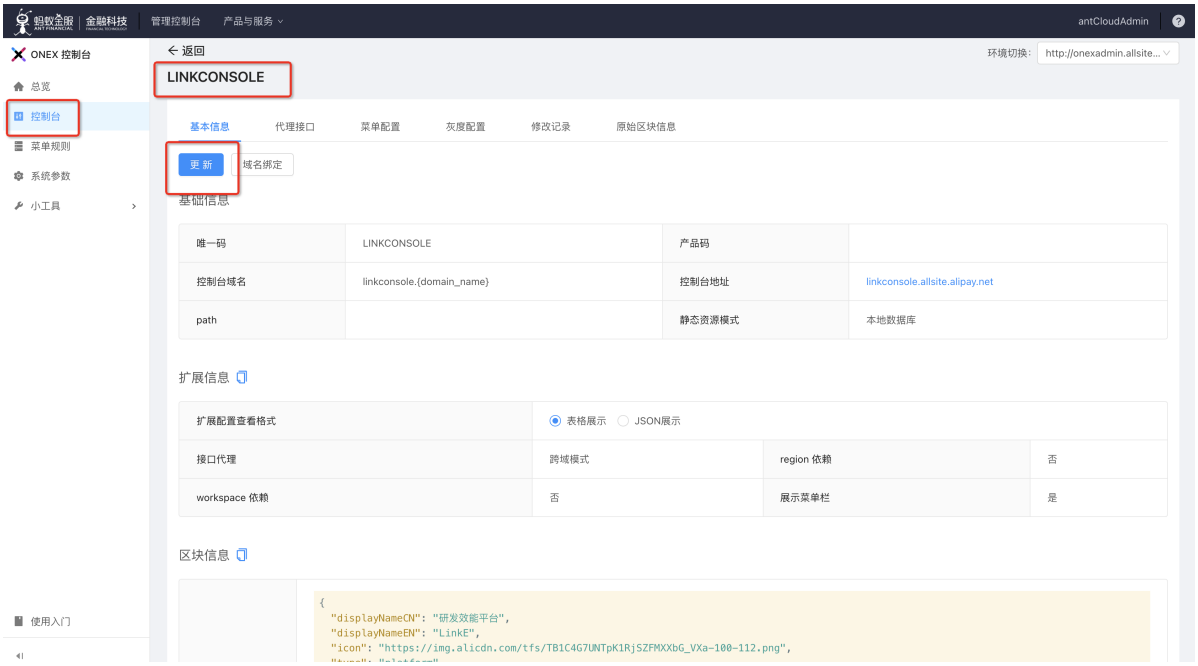
> 说明
>
> - 参数 `${env.info.domain}` 为域名配置，假设客户环境云游部署域名是 yunyou.testcompany.com，则域名配置变量是 `${env.info.domain} = testcompany.com` 。
> - 假设客户环境的部署域名是 testcompany.com，则 LinkE 访问入口为 linkconsole.testcompany.com，用户使用命令行推送代码的入口是 code.testcompany.com。

6.2. OneConsole 获取前端资源方式

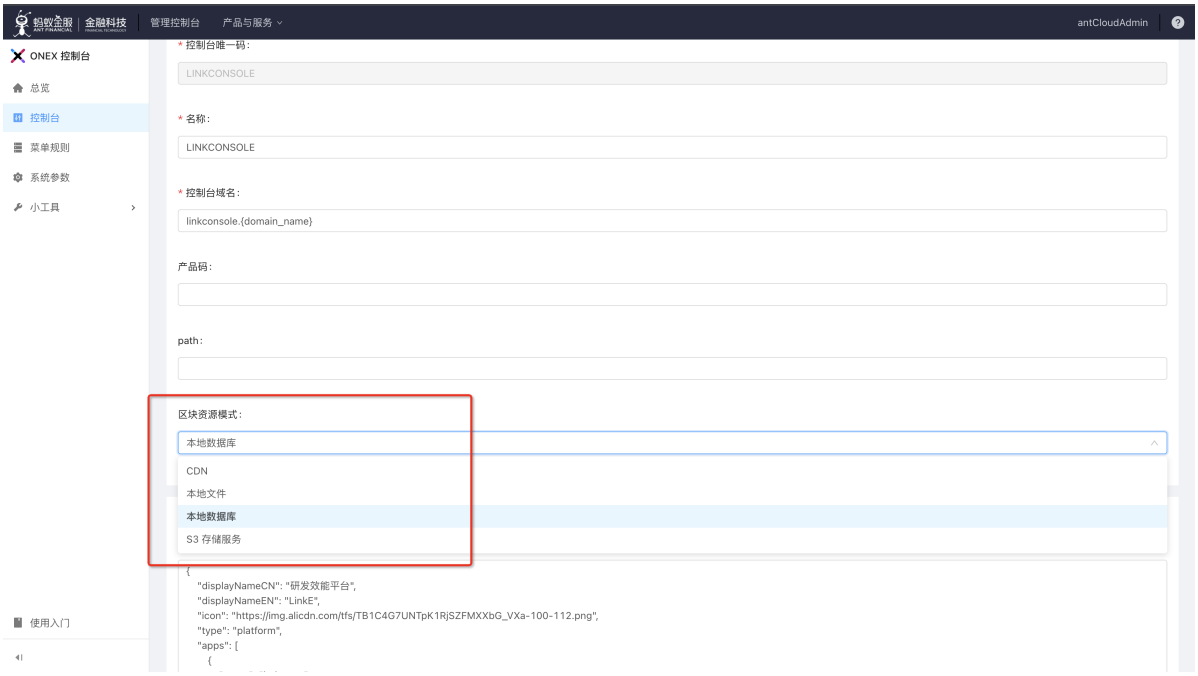
> 说明
>
> - LinkE 2211 及以后版本无需配置 OneConsole，可跳过该文档。
> - 2211 之前的版本需要根据此文档配置 OneConsole。

区块配置

1. 登录 ONEX 控制台，找到 LINKCONSOLE。
2. 单击 **更新**。



3. 找到区块资源模式一栏。

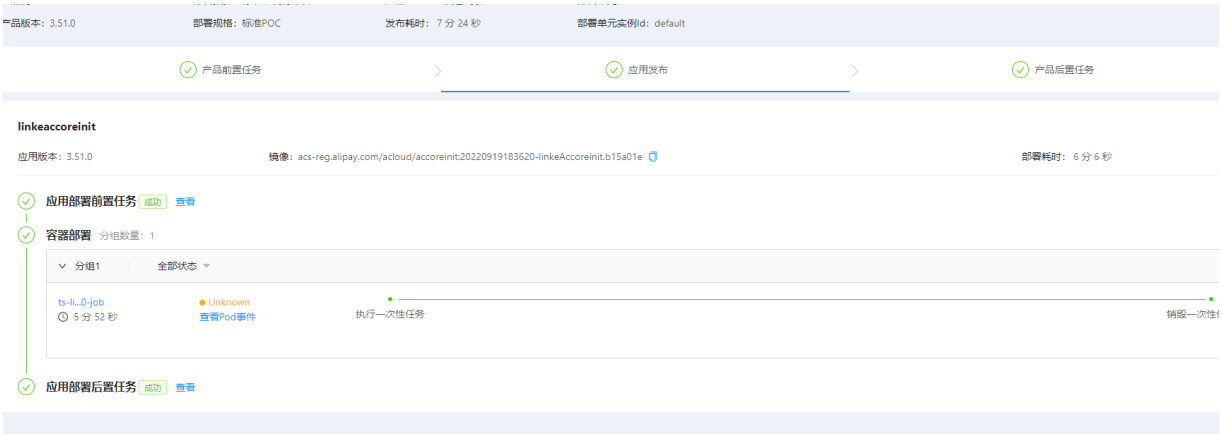


4. 修改区块资源模式，选择 **本地文件**。
5. 在页面最下方单击 **保存** 即可生效。

7.部署验证

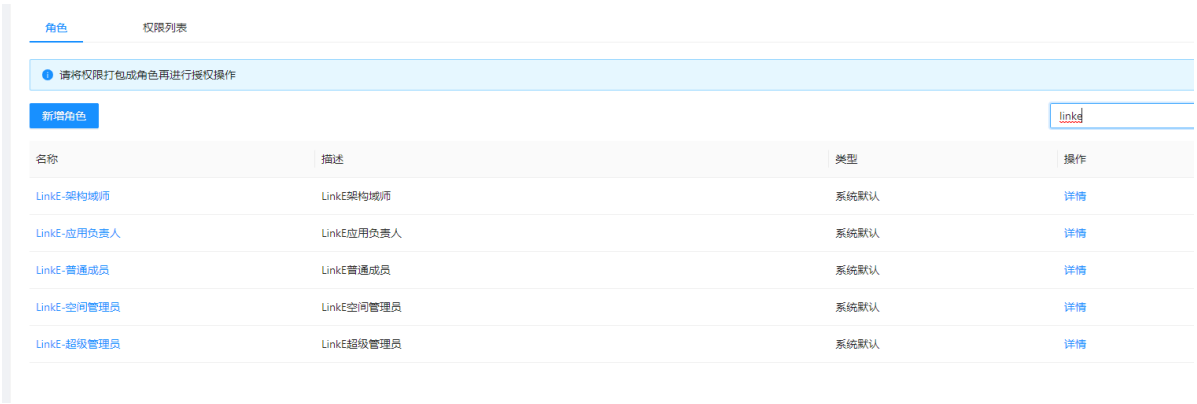
7.1. 确认 IAM 录入权限角色

确认一次性任务 linkeaccoreinit 应成功执行，如下图所示：



执行完成后，进入 IAM 控制台，确认已新增如下预置权限与角色：

- LinkE 预置角色



- LinkE 预置权限

7.2. 执行自动化测试

完成 LinkE 前后端的部署后，您可以按照本文内容对 LinkE 进行自动化测试配置。

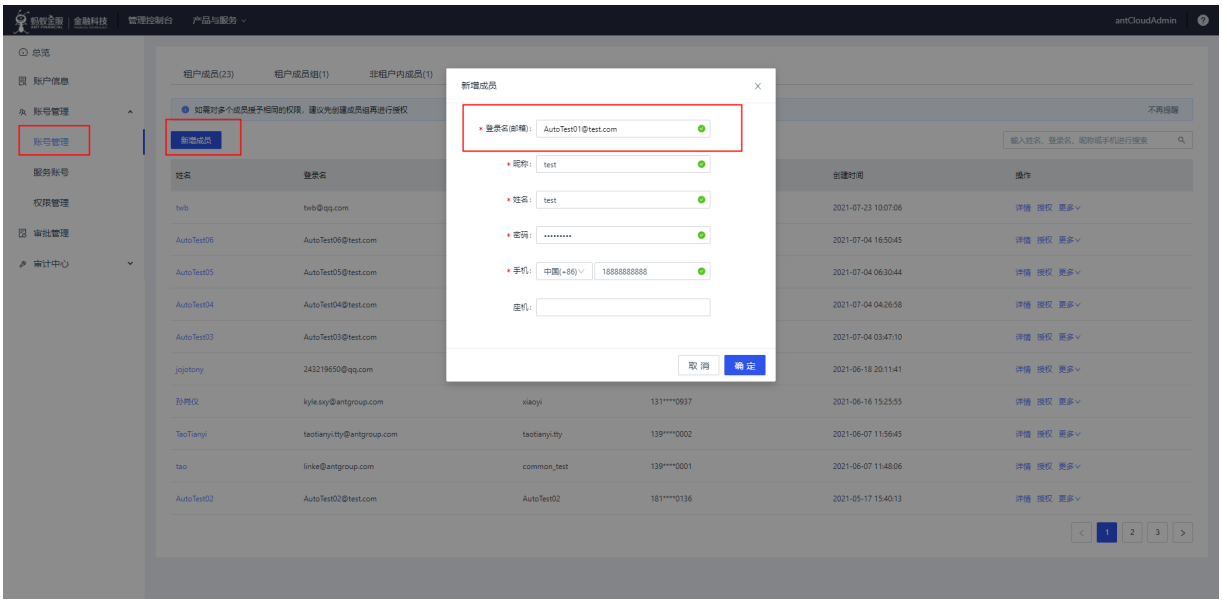
自动化测试配置有 3 个方面：初始化用户、LINKEPORTAL、ANTCODE，下文分别介绍相应的操作步骤。

初始化用户

步骤一：登录 IAM 平台并注册两个新用户

两个新用户账号分别为 `AutoTest01@test.com` 和 `AutoTest02@test.com`，其他信息任意填写，密码为：`antCloud123`。

> 说明
>
> 如果已经注册，无需重新注册。



步骤二：使用新账号登录 LinkE

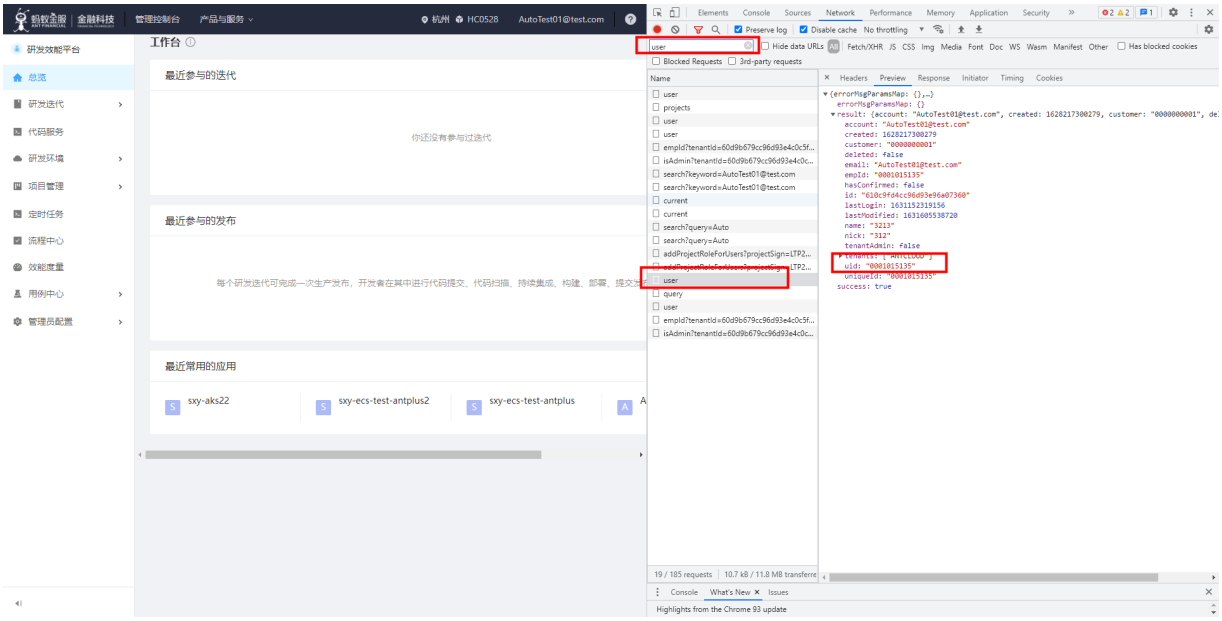
注册完成后，使用以下 3 个账号分别登录 LinkE。

- AutoTest01@test.com
- AutoTest02@test.com
- antCloudAdmin

> 重要
>
> 此步骤必须进行，否则自动化配置无效！

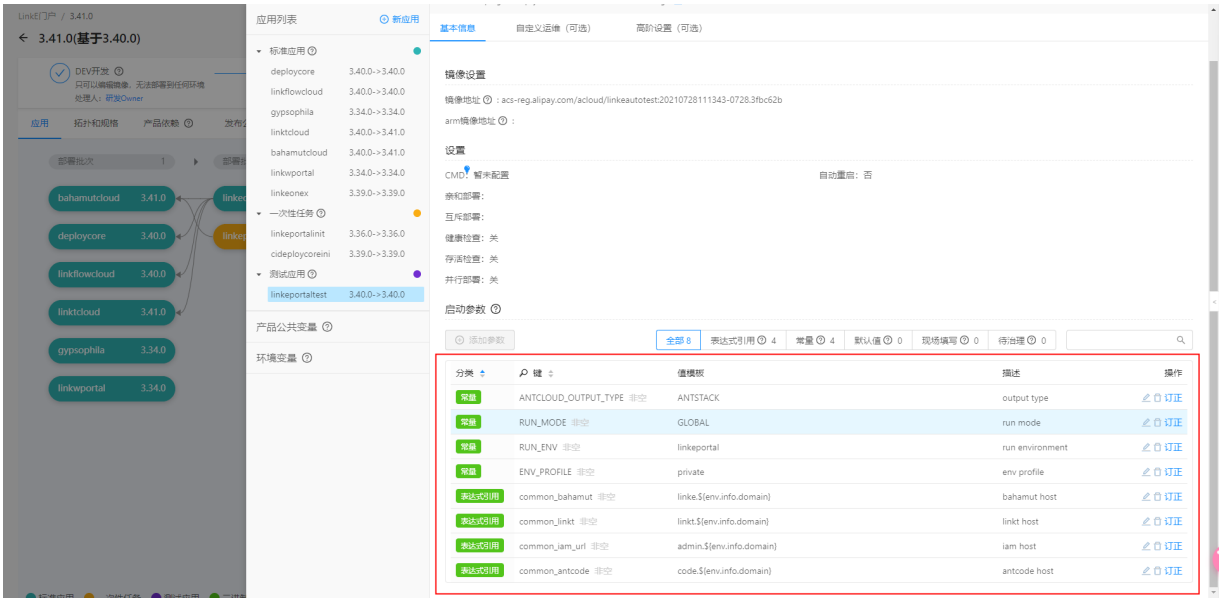
步骤三：获取 AutoTest01@test.com 账号 ID

使用 AutoTest01@test.com 登录 LinkE 时，打开浏览器开发者工具，进入到 network 这一栏，过滤 `user` 字段，可得到如下图所示接口，展开 preview 这一栏，找到 `uid` 字段即可。



LINKEPORTAL

登录云游平台， 进入 LINKEPORTAL 应用页面，查看/修改测试应用 **linkeportaltest** 的启动参数信息，参数说明如下表所示。



键	值模板	描述	是否现场填写
access_key	默认自动渲染（来源 IAM 控制台 LinkE 使用的租户管理员服务号）。	用户 AK，下游调用 IAM 做鉴权时，所用的 AK（ROC 租户管理员服务号，不然有权限问题）。	是（填写在产品公共参数可继承）
secret_key	默认自动渲染（来源 IAM 控制台 LinkE 使用的租户管理员服务号）。	用户 SK，下游调用 IAM 做鉴权时，所用的 SK（ROC 租户管理员服务号，不然有权限问题）。	是（填写在产品公共参数里可继承）

ANTCLOUD_OUTPUT_TYPE	ANTSTACK	output type	否
RUN_MODE	GLOBAL	run mode	否
RUN_ENV	linkeportal	run environment	否
ENV_PROFILE	private	env profile	否
common_bahamut	linkeapi.${env.info.domain}	bahamut host	否
common_linkt	linkeapi.${env.info.domain}	linkt host	否
common_iam_url	admin.${env.info.domain}	iam host	否
common_antcode	code.${env.info.domain}	antcode host	否
common_linkflow	linkeapi.${env.info.domain}	linkflow host	否
common_linke	linkconsole.${env.info.domain}	linke web	否
common_user1	AutoTest01@test.com	在初始化用户中，AutoTest01@test.com 账号 id。	否
common_user2	AutoTest02@test.com	在初始化用户中，AutoTest02@test.com 账号 id。	否
common_user2ID		user2 用户 ID	是
common_schema	http:// 或者 https://	HTTP 协议头	否

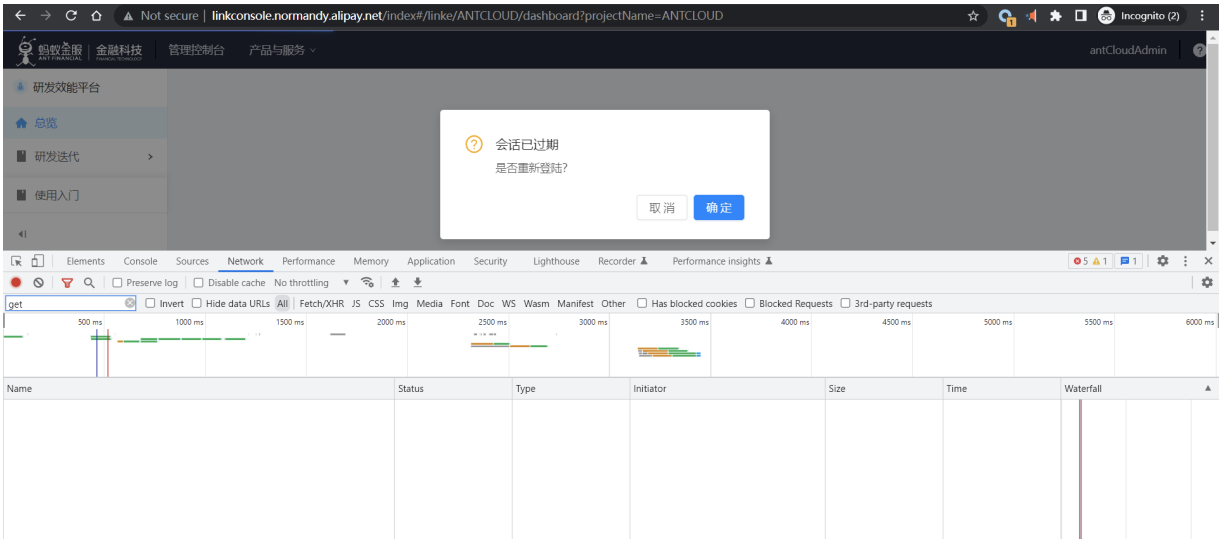
云游 Pod 运行结果

8.部署常见问题

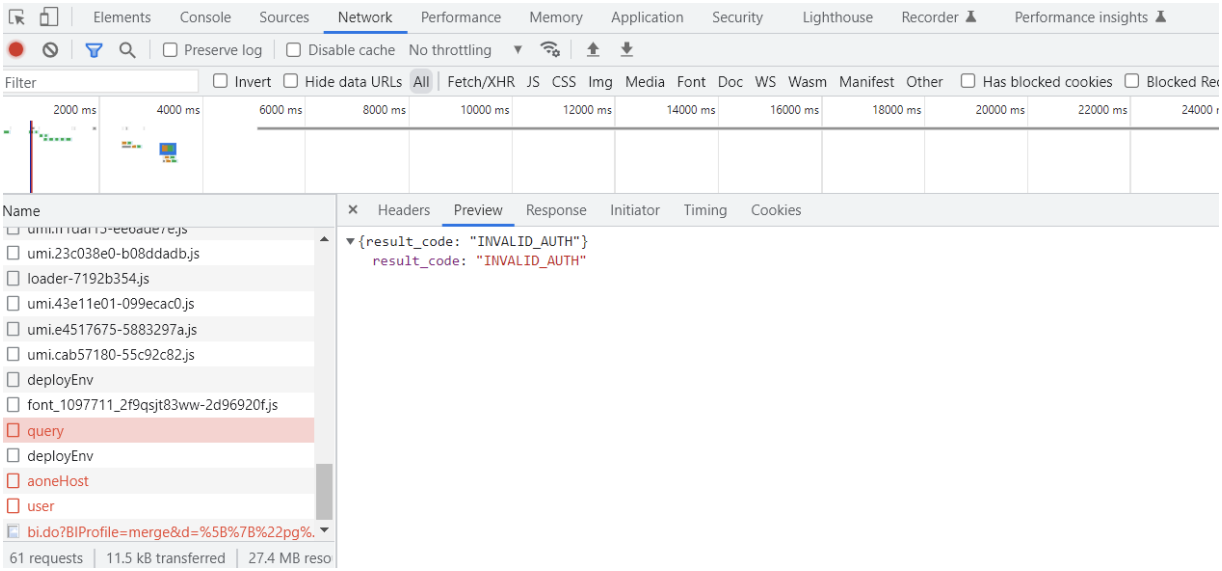
8.1. LinkE 产品完成部署后账号无法登录

问题现象

部署完成后，进入 LinkE 首页，一直提示“**会话已过期**”。



浏览器控制台报 API response 为 **INVALID_AUTH**。



问题原因

LinkE 在部署前，需要在 IAM ROC 租户下，创手动创建一个 LINKE 专用租户管理员服务账号。创建完成后，需要将其 AK/SK 填至 LINKEBASE 产品公共参数中。

此服务账号具有 ROC 租户下管理员权限，而 ROC 是 IAM 内部高权限租户，因此此服务器拥有当前环境所有租户的管理员权限，被 LinkE 产品用于查询各租户名下的应用、应用服务列表，并校验浏览器登录态等功能。

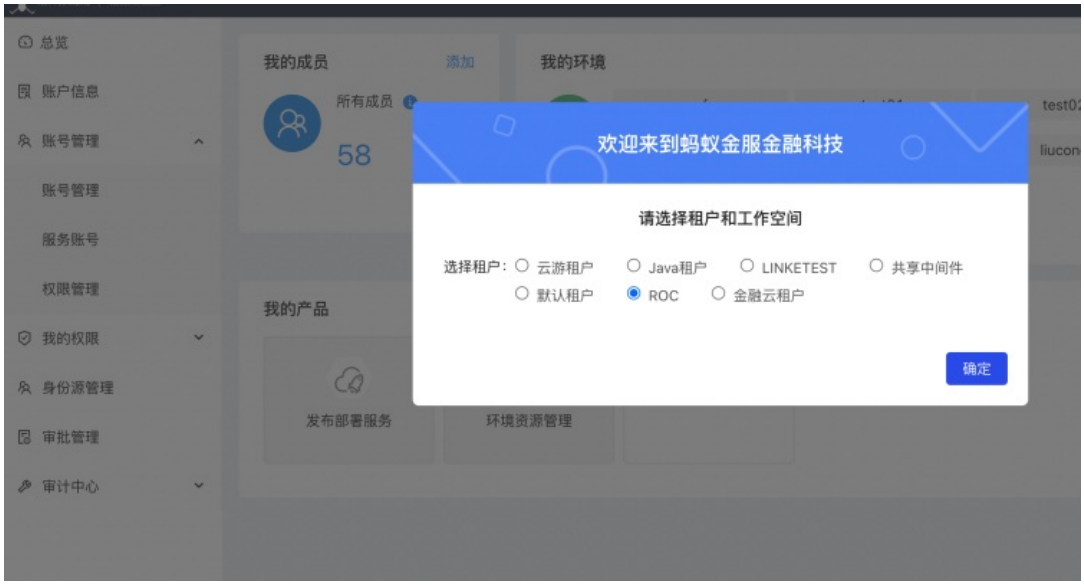
应用 LinkU 第一次启动时，会尝试将该 AK/SK 写入 linku_cloud 数据库的 linku_secret_config 数据表。若写入不成功，则会阻塞 LinkE 产品的用户登录态查询请求。

解决方案

如果数据不符合预期，则需订正 LinkU 应用的数据库。

1. LinkE 专用服务号的信息查询

   i. 登录 IAM 控制台切换到 ROC 租户。



   ii. 查看 LinkE 服务号信息。

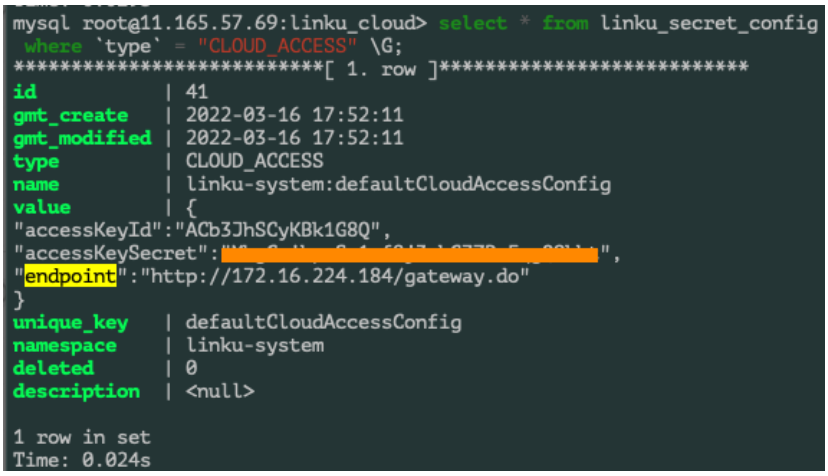
> 说明
> 需要是租户管理员权限，没有则需要手动创建。

2. 应用 LinkU 数据库已保存信息的查询

执行如下命令登录 LINKBASE 产品使用的 MySQL 数据库：

```
use linku_cloud;

select * from linku_secret_config where `type` = "CLOUD_ACCESS" \G;
```

如下图所示，默认 linku_secret_config 中会有一条 IAM 中 LinkE 服务号 AK/SK 的记录，LinkE 会使用此 AK/SK 校验用户的登录态。



> 说明
>
> 上图中的 endpoint 为 OP 产品应用 acprodapigw 对应的负载均衡 OP-antprodgw_internal_lb 地址。

3. 手动订库

执行如下命令：

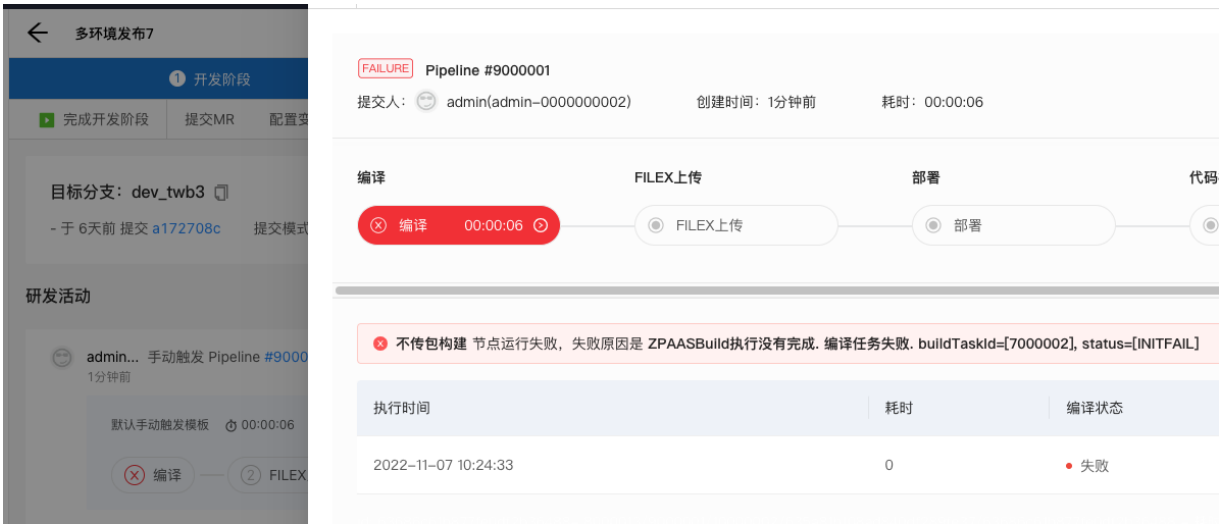
```
-- 新 value 值
set @opvalue = '{
  "accessKeyId":"ak",
  "accessKeySecret":"sk",
  "endpoint":"endpoint"
}';
-- 原 op ak/sk 记录 id
set @origin_id = -1;

-- 更新当前记录
update linku_secret_config set `value` = @opvalue
where `type` = "CLOUD_ACCESS" and id = @origin_id;

-- 或者插入一条新记录
-- insert into linku_secret_config ...
```

> 说明
>
> 其中 AK/SK 需要在 IAM 控制台上获取。

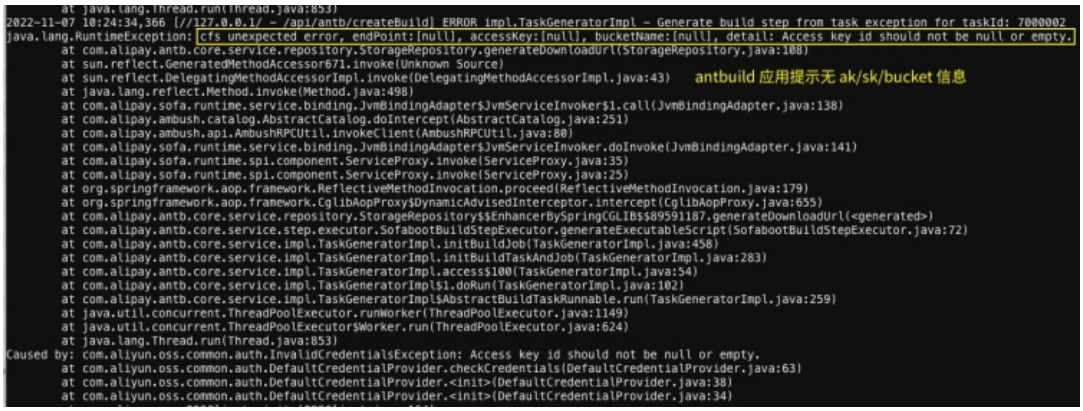
8.2. LinkE部署后经典编译无法运行

问题现象

经典编译无法运行，页面如下图所示：



云游 antbuild 应用日志提示：



解决方案

登录 LinkU 数据库查看 linku_secret_config 表。

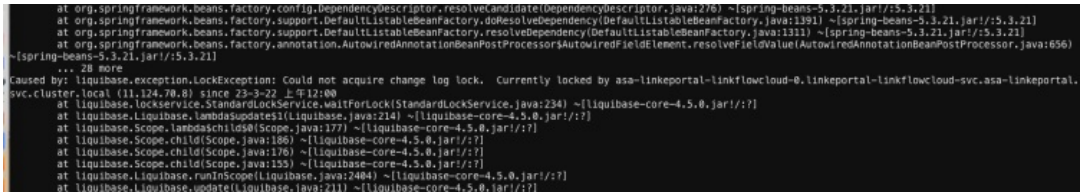
```
select * from linku_secret_config where `type` = "OSS" \G;
```

查询是否存在 `name="antb:ossConfig"` 的数据行，若没有则手动插入一条。

8.3. LinkFlow 启动报错 - databasechangeloglock

问题现象

LinkFlow 启动时报错，如下图所示：

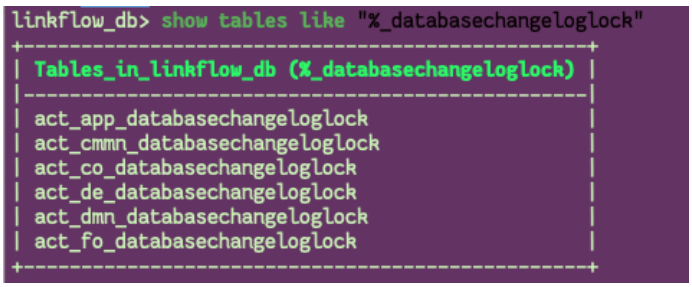


解决方案

1. 登录 LinkFlow 数据库。

```
show tables like "%_databasechangeloglock"
```

2. 查看以 `_databasechangeloglock` 结尾的数据表。



3. 查看数据表，执行如下命令删除 lockedBy 错误日志记录的一行。

```
delete from act_dmn_databasechangeloglock where id = 1
```

如下图所示：

8.4. 删除在 LinkE 数据库中已经初始化的租户信息

当租户初始化失败时，需要清除在 LinkE 数据库中已经初始化的租户信息。

问题场景

租户名下的用户第一次登录 LinkE，在管理员配置中没有发现预置迭代模板与流水线列表，表示租户初始化失败。

常见于 LinkU、LinkM 使用的数据库信息有误，或者多次重新部署导致 ANT CLOUD 此预置金融云租户数据异常。

解决方案

已初始化的租户存储于 bahamut、LinkU、LinkM 三个应用的数据库，需要依次执行以下命令进行清除。

- in linke mongodb

```
use bahamut;

// 重命名表作备份
db.tenant.renameCollection("backup_tenant");
db.cloudTenant.renameCollection("backup_cloudTenant");

// 删除表
// db.tenant.drop()
// db.cloudTenant.drop()
```

- in linke mysql

```
-- linku 数据清除
use linku_cloud;

select * from linku_tp_auth;

truncate table linku_tp_auth;


-- linkm 数据清除
use metacenter_cloud;

select * from metacenter_tenant;

truncate table metacenter_tenant;
```