SOFAStack

研发效能平台 LinkE

部署指南

产品版本：AntStack Plus 1.11.0

文档版本：20220928

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。

模块	说明
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。

模块	说明
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

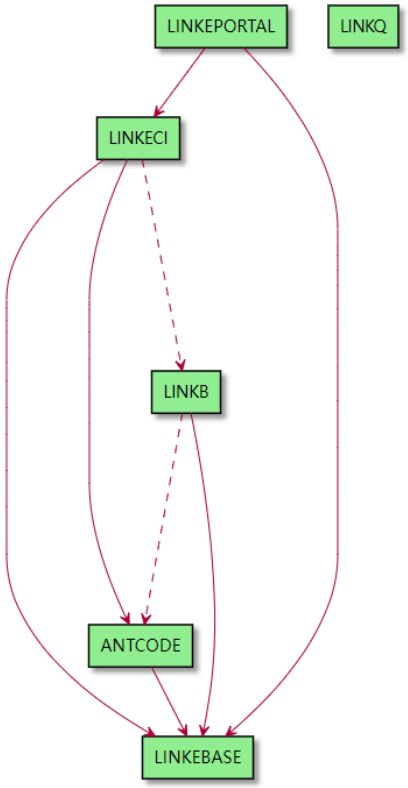
  默认 bucket 为 `linkestorage-linkestorage`

- 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。

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.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 + ANT CODE + 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。

4.部署准备

4.1. 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

产品	应用	服务器类型	CPU	内存	硬盘	数量（台）
	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
2	4	1000	2	4	8	2000

CPU 规格	内存规格	磁盘规格	实例数量	CPU	内存	磁盘
2	4	500	3	6	12	1500
2	4	100	44	88	176	4400
基础部署包	不涉及	不涉及	不涉及	114	228	8700
2	4	100	5	10	20	500
可选部署包	不涉及	不涉及	不涉及	124	248	9200

4.2. 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。

> **注意**
>
> 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。

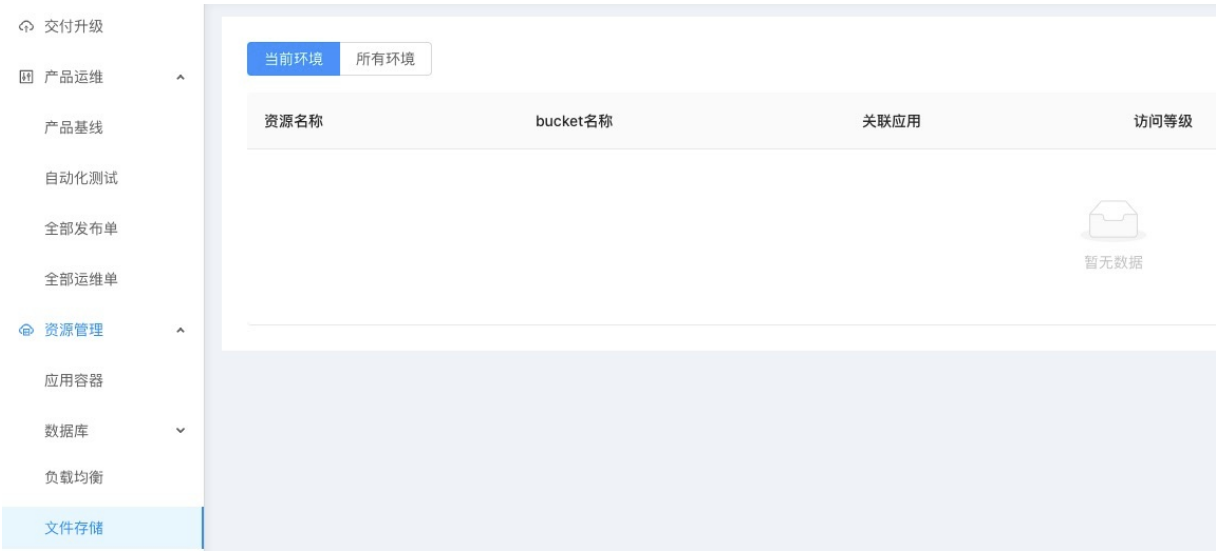
```
LINKEBASE -- logstore
privateEndpoint: xxx.xxx.xxx/
publicEndpoint: http://xxx.xxx.xxx/
bucketName: logstore-logstore
accessKey:
accessSecret:

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` :

4.3. 数据库资源

数据库名称（schema）	数据库用户名（同数据库名）	连接密码	默认所属物理实例（资源角色名）
linku_cloud	linku_cloud	默认值	linke_base
fabric_cloud	fabric_cloud	默认值	linke_base
metacenter_cloud	metacenter_cloud	默认值	linke_base
linklog	linklog	默认值	linke_base
antcode	antcode	默认值	antcode_db
batmanshard	batmanshard	默认值	antcode_db
codecenter	codecenter	默认值	antcode_db
linkb_db_01	linkb_db_01	默认值	linkb_db_01
core_cloud	core_cloud	默认值	aclinkecore
linkt_db	linkt_db	默认值	linkt_db
deploycore_db	deploycore_db	默认值	linkt_db

数据库名称（schema）	数据库用户名（同数据库名）	连接密码	默认所属物理实例（资源角色名）
linkflow_db	linkflow_db	默认值	linkt_db
easycase_apps	easycase_apps	默认值	linkt_db
amock_db	amock_db	默认值	amock_db
aimsdb	aimsdb	默认值	aimsdb

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：

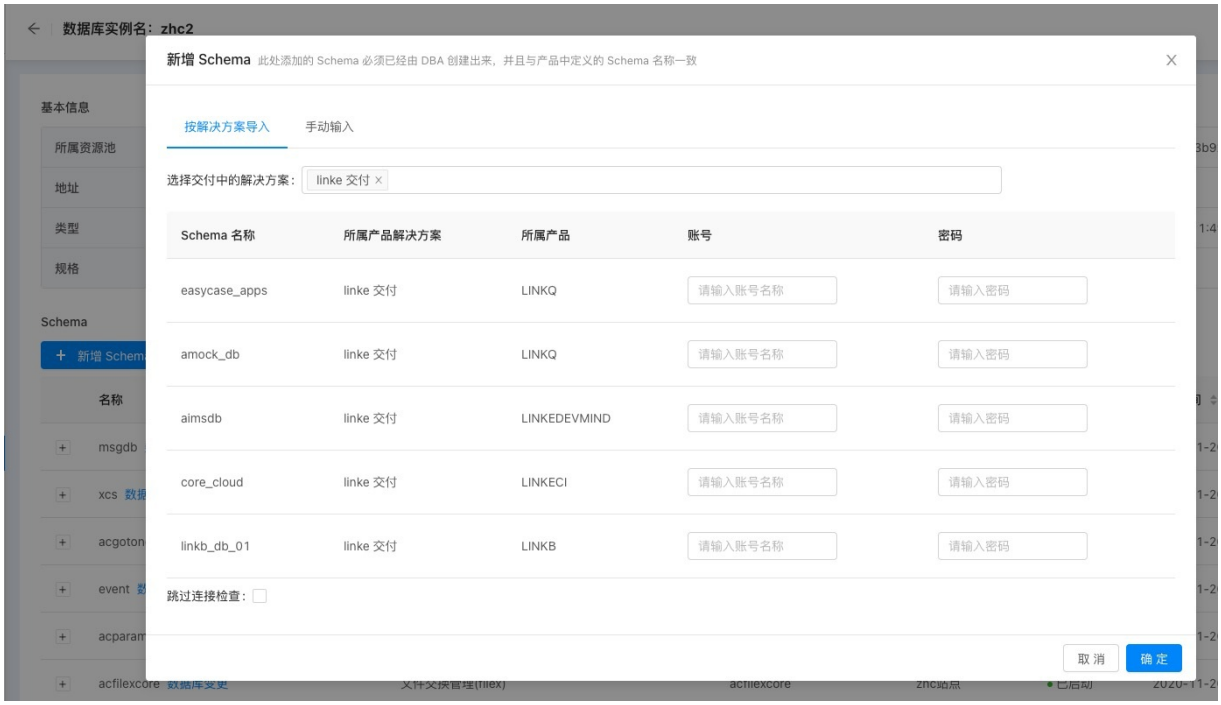
```
create schema linku_cloud;
create schema fabric_cloud;
create schema metacenter_cloud;
create schema linklog;
create schema antcode;
create schema batmanshard;
create schema codecenter;
create schema linkb_db_01;
create schema core_cloud;
create schema linkt_db;
create schema deploycore_db;
create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

create user 'linku_cloud'@'%' identified by 'PaaS123456';
create user 'fabric_cloud'@'%' identified by 'PaaS123456';
create user 'metacenter_cloud'@'%' identified by 'PaaS123456';
create user 'linklog'@'%' identified by 'PaaS123456';
create user 'antcode'@'%' identified by 'PaaS123456';
create user 'batmanshard'@'%' identified by 'PaaS123456';
create user 'codecenter'@'%' identified by 'PaaS123456';
create user 'linkb_db_01'@'%' identified by 'PaaS123456';
create user 'core_cloud'@'%' identified by 'PaaS123456';
create user 'linkt_db'@'%' identified by 'PaaS123456';
create user 'deploycore_db'@'%' identified by 'PaaS123456';
create user 'linkflow_db'@'%' identified by 'PaaS123456';

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

-- 如果数据库是 mysql 则还需要为 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;
alter table ACT_FO_FORM_DEFINITION character set utf8mb4;
alter table ACT_FO_FORM_DEFINITION modify NAME_ varchar(50) character set utf8mb4;
```

> **说明**
>
> 可按需创建，未部署的应用可不创建，如 amock 不部署则可不创建 amock 的数据库信息。

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。

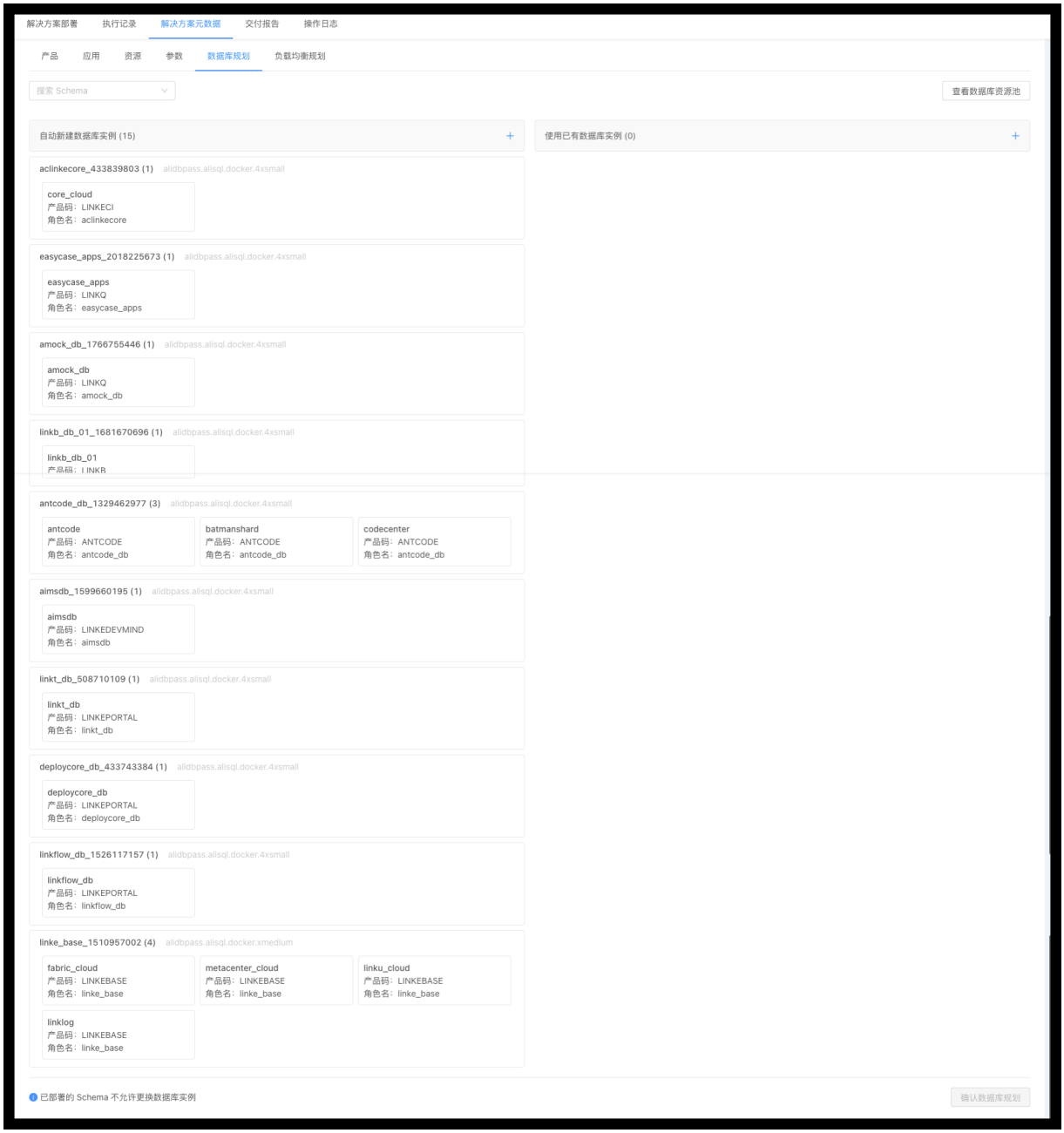
> **说明**
>
> 账号和 Schema 名称一致，密码按照创建账号时使用的密码，默认均使用 LinkE 默认密码：`PaaS123456`。



规划数据库

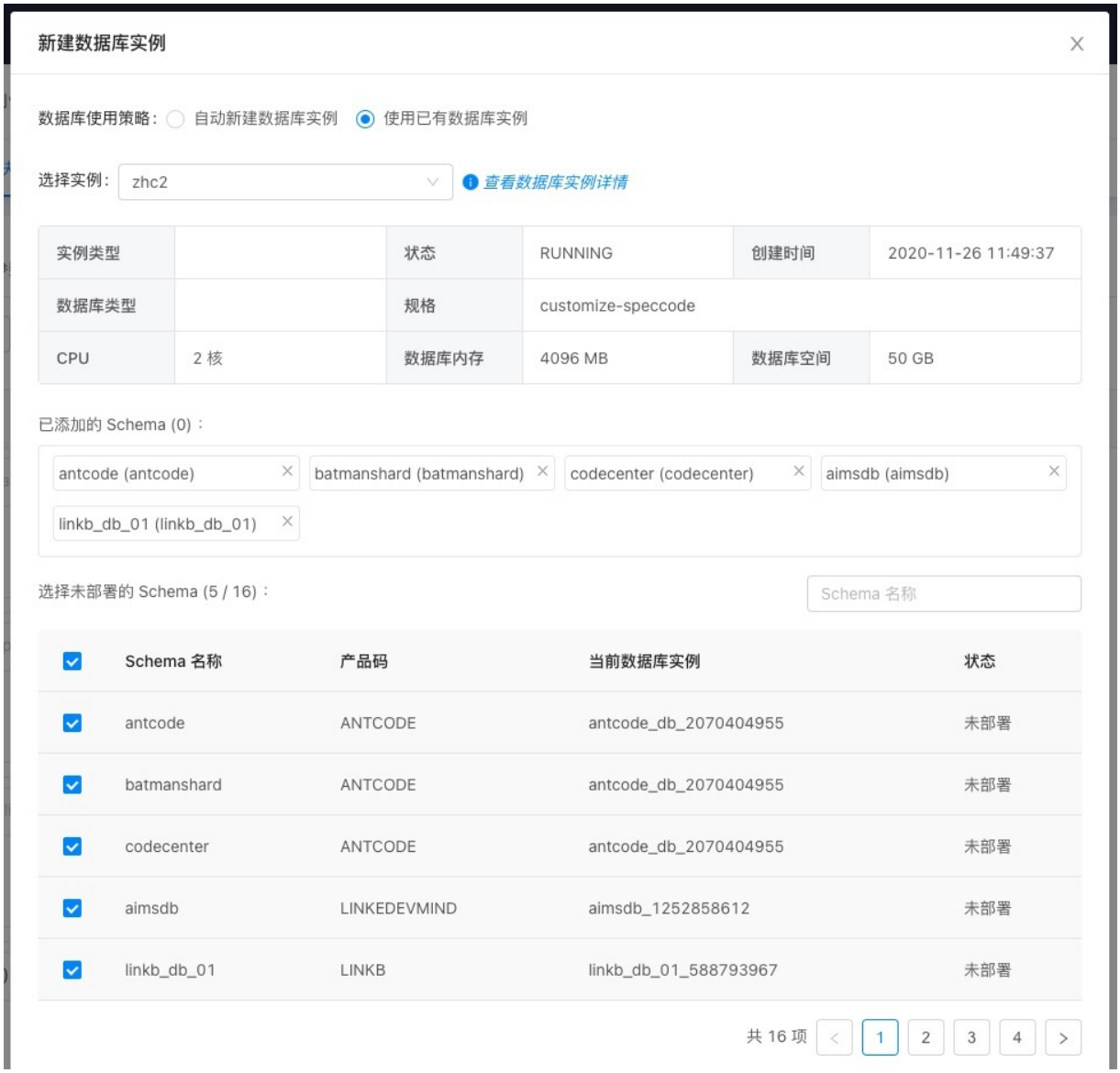
根据 Local 环境选择新建实例还是复用实例，自动创建数据的情况直接选择新建，复用实例则添加已创建的已有的实例。

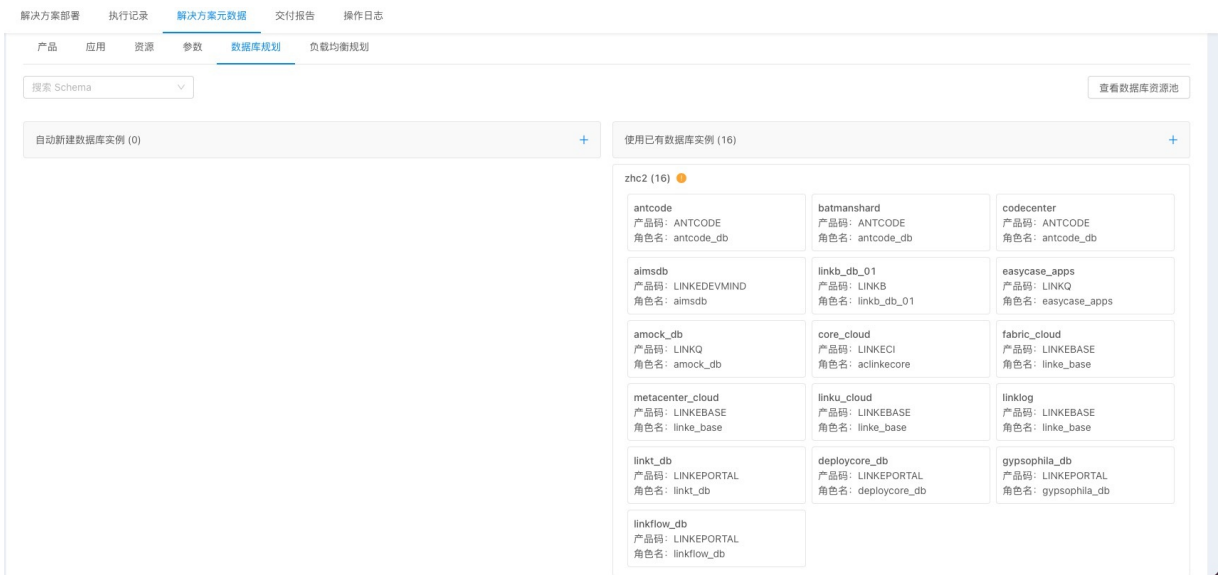
选择自动新建数据库实例



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

4.4. 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。

4.5. SidecarSet

暂无配置

5.后端应用部署

5.1. 产品公共参数

由于存储资源类型根据底座不同有所差异，所以需要设置 **STORAGE_TYPE** 参数区分存储类型。同时需要修改所有产品的 **ACCESS_KEY 和 ACCESS_SECRET** 为创建好的统一的存储空间调用 AK 和 SK，并设置 endpoint 为实际的统一存储 endpoint 地址。

- 底座（阿里云）存储为 OSS 则填写 `oss`，底座（AntStack Plus）存储为 minio/s3 时则填写 `s3`。

- 底座存储为 OSS 时直接填写调用域名（不需要以 http:// 开头），底座存储为 S3 时请填写 AFS 的 serverUrl（需要以 http:// 开头），例如 `http://afsserver.antstack-plus.net:8080`。

- 底座存储使用的 AK/SK 可以从阿里云控制台或者 minio 搭建者处获取（通常与金融云账号 AK 和 SK 不同）。

5.2. LINKEBASE

5.2.1. 部署说明

> **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。

应用配置规划

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），通过云游根据不同底座将容器内数据路径挂载出来（宿主机 volume 或者 pvc，根据云游而定）。

- 在 plus 底座挂载 pvc，则无需关心具体容器落在那个 node 上
- 如果是非 plus 底座挂载宿主机 volume，就必须考虑具体 node 的问题，linkemongo 所挂载的宿主机目录为 `/home/t4/data/mongo`。
  - 宿主机默认挂载 `/home/t4` 数据盘的情况下，可以直接云游发布部署。
  - 宿主机没有挂载 `/home/t4` 数据盘的情况下，必须要为 linkemongo 单独挂载数据盘到 `/home/t4` 路径（三台 mongo 分别指定三个 node 挂载 `/home/t4` 数据盘），然后通过云游的配置规划里的节点选择器指定到挂载了数据盘的 node 上再进行发布部署。

    > **说明**
    >
    > 老版本云游采用的是固定 node 的策略，因为理论上来说数据持久化的情况下不能出现节点漂移，对于 linkemongo 来说数据是自动同步到每个节点上的所以理论上来说在固定的三个机器范围内漂流也没有问题。

linkenexus

> **注意**
>
> 不建议部署，客户通常有自建的 Nexus 仓库，如果需要部署 Nexus 的拓扑，由于 Nexus 也是有状态应用，需要持久化数据，需同 Mongo 一样进行配置规划。

数据备份

为 linkemongo 容器对应的所挂载数据磁盘或者 pvc 进行备份配置。

> **注意**
>
> 如果是 plus 环境请务必确认数据落在 pvc 上，如果不是则要确认部署环境是否正常打标 plus 环境且要打标后重新部署。

- 阿里云底座直接创建磁盘自动快照策略（默认可以每日凌晨 3 点备份保留 30 天），然后为三个 linkemongo 的数据盘设置该备份策略。
- 物理机器依据物理机器本身的数据备份策略。

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，init linkebasedb 的一次性任务（job）必须成功。

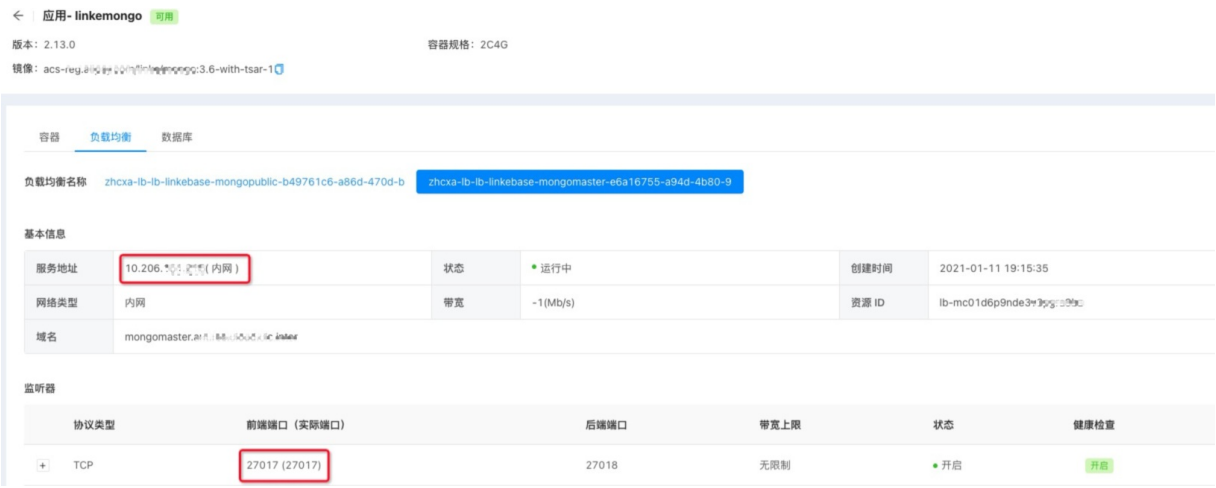
5.2.3. 部署检查

mongo 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
```



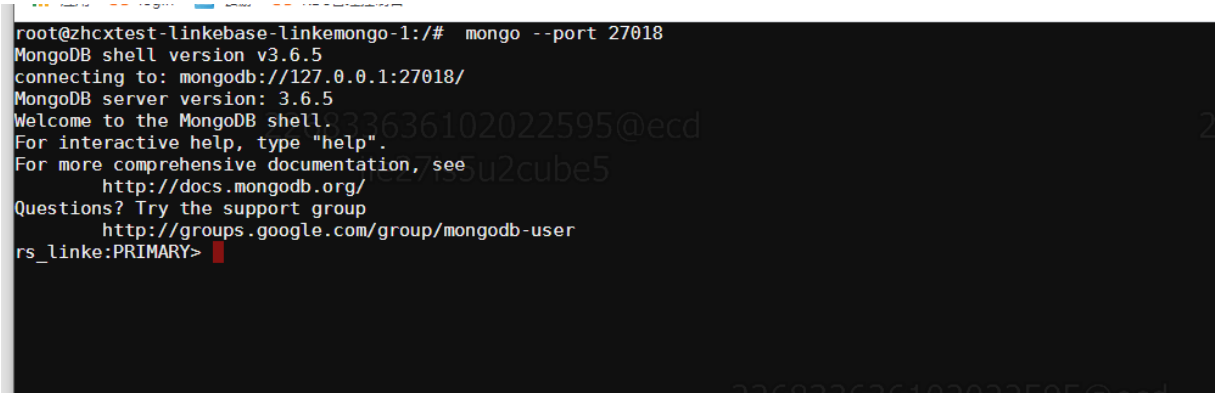
```
rs_linke:PRIMARY> use admin;
switched to db admin
rs_linke:PRIMARY> db.auth('linke','LinkE2017');
1
rs_linke:PRIMARY> rs.status()
{
        "set" : "rs_linke",
        "date" : ISODate("2021-03-02T10:11:10.122Z"),
        "myState" : 1,
        "term" : NumberLong(4),
        "heartbeatIntervalMillis" : 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` ，预计进入 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。



5.3. ANTCODE

5.3.1. 部署说明

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

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），通过云游根据不同底座将容器内数据路径挂载出来（宿主机 volume 或者 pvc，根据云游而定）。

- 如果是 plus 底座挂载 pvc，则无需关心具体容器落在那个 node 上。
- 如果是非 plus 底座挂载宿主机 volume，就务必需要考虑具体 node 的问题，antcodenode 所挂载的宿主机目录为 /home/t4/data/repo。
  - 宿主机默认挂载 `/home/t4` 数据盘的情况下，可以直接云游发布部署。
  - 宿主机没有挂载 `/home/t4` 数据盘的情况下，必须要为 antcodenode 单独挂载数据盘到 `/home/t4` 路径（两台 antcodenode 分别指定两个 node 挂载 `/home/t4` 数据盘），然后通过云游的配置规划里的节点选择器指定到挂载了数据盘的 node 上再进行发布部署。

    > 说明
    >
    > 老版本云游采用的是固定 node 的策略，因为理论上来说数据持久化的情况下不能出现节点漂移，对于 antcodenode 来说数据是自动同步到每个节点上的所以理论上来说在固定的三个机器范围内漂流也没有问题。

数据备份

为 antcodenode 容器对应的所挂载数据磁盘或者 pvc 进行备份配置。

- 阿里云底座直接创建磁盘自动快照策略（默认可以每日凌晨 3 点备份保留 30 天），然后为三个 antcodenode 的数据盘设置该备份策略。
- 物理机器依据物理机器本身的数据备份策略。

5.3.2. 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。

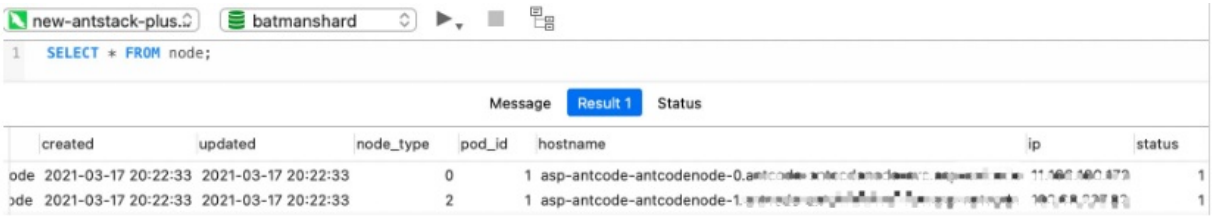
> **说明**
>
> 在进行应用初始化时，batmandatainit（1.34 版本以后不需要）、antcodedatainit、codecenterinit 的一次性任务（job）必须成功。

5.3.3. 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;
```



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。

5.4. LINKB

5.4.1. 部署说明

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

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，目前云游的不同版本底座机器上启用的容器服务不同，LINKEB 需要根据底座情况进行部署调整。

- 先直接通过云游发布部署 LINKB 产品。
- 可能需要手动调整的应用为 buildrunner。

5.4.2. 部署方式

云游全局 Docker 部署方案

底座宿主机全部为标准 Docker 服务时，可以直接使用云游部署后的服务，不需要做调整。因为 buildrunner 是通过 host volume 来挂载宿主机的 Docker 服务和配置从而在容器内使用 Docker 来完成镜像构建、推送等功能的，但要求宿主机的 Docker 必须满足挂载需求。

- 宿主机上的 Docker 程序为 `/usr/bin/docker` 。
- 宿主机上启动了 Dockerd 进程，且使用的是默认监听的 Unix 域套接字，路径为 `/var/run/docker.sock` 。

```
#ps -ef | grep sock
root      80163  74811  0 17:12 pts/0    00:00:00 grep --color=auto sock
root     130981      1  0 4月15 ?       00:15:17 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --data-root=/home/data/docker --registry-mirror https://cl6o4b1o.mirror.aliyuncs.com --registry-mirror https://hub-mirror.c.163.com --registry-mirror https://docker.mirrors.ustc.edu.cn --insecure-registry hub.antstack-plus.net

#ps -ef | grep dockerd
root      80174  74811  0 17:12 pts/0    00:00:00 grep --color=auto dockerd
root     130981      1  0 4月15 ?       00:15:17 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --data-root=/home/data/docker --registry-mirror https://cl6o4b1o.mirror.aliyuncs.com --registry-mirror https://hub-mirror.c.163.com --registry-mirror https://docker.mirrors.ustc.edu.cn --insecure-registry hub.antstack-plus.net

#ll /var/run/docker
docker/      docker.pid   docker.sock

#ll /var/run/docker.sock
srw-rw---- 1 root docker 0 4月  15 13:18 /var/run/docker.sock

#whereis docker
docker: /usr/bin/docker /etc/docker /usr/share/man/man1/docker.1.gz
```

云游部分 Docker 部署方案

云游底座上部分机器提供 Docker 服务（如标准机器只提供 containerd，ops 机器提供 dockerd），此时可以通过将 buildrunner 容器调度到对应的 Docker 宿主机上进行部署，有两种方式可以做到：

- 部署 buildrunner 时（或者部署完删除重新部署时）将非 Docker 的机器暂停调度，只保留 Docker 的机器进行调度使得 buildrunner 部署到 Docker 的宿主机上。
- 需要在 captain 中为含 docker 的宿主机添加类似于 `docker.io/used: "true"`，然后修改 StafulSet 的 yaml 为其在 spec 配置中添加 nodeSelector 筛选打标的宿主机。

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: tomcat-deploy
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: tomcat-app
    spec:
      nodeSelector:
        cloudnil.com/role: dev #指定调度节点为带有label标记为：cloudnil.com/role=dev的node节点
      containers:
      - name: tomcat
        image: tomcat:8.0
        ports:
        - containerPort: 8080
```

Docker 手动部署方案

环境中提供一台单独的物理机，安装 Docker 服务并启动 Dockerd 同 云游全局 Docker 部署方案 一致，然后需要手动拉起 buildrunner 容器：

- 先进入云游部署的 buildrunner 容器，执行 env 将容器内的环境变量拷贝出来。
- 运行 buildrunner 的镜像，docker run 时将上一步拷贝出来的环境变量通过 `-e` 挂上，同时设置 volume 的挂载。

```
--mount type=bind,source=/var/run/docker.sock,target=/var/run/docker.sock --mount type=bind
,source=/usr/bin/docker,target=/usr/bin/docker --mount type=bind,source=/etc/hosts,target=/
etc/hosts
```

BuildKit 手动部署方案

AntStack Plus 的底座机器提供的是 containerd 服务，由于目前云游底座的宿主机内核版本限制以及云游创建应用的 annotations 功能不完善，暂不支持云游自动部署 buildkit 的构建容器。需要手动通过一台物理机部署 buildkitd 服务为 LinkE 提供镜像构建能力。

安装 buildkitd：

```
wget https://deploying-tools.oss-cn-hangzhou.aliyuncs.com/linke/solutions/buildkitd-v0.8.2-antcloud.zip

# 部署在这个文件夹里
mkdir /opt/local
unzip buildkitd-v0.8.2-antcloud.zip -d /opt/local
cd /opt/local

cd /opt/local
sh makedirs-for-buildkitd.sh
export PATH=$PATH:/opt/local/bin
```

- 编辑系统配置文件 `/etc/hosts`，一般需要添加 hub 镜像中心域名的 host。
- 编辑 buildkitd 配置文件 `/opt/local/etc/buildkitd.toml`，添加 insecure 镜像源的配置（用于 pull 镜像）。

> **说明**
>
> 如果自建的镜像仓库已配置过 SSL 证书，则可以略过此步配置。

```
[registry."hub.antplus.com"]
  http = true
  insecure = true
```

```
#cat /opt/local/etc/buildkitd.toml
insecure-entitlements = [ "network.host", "security.insecure" ]

[worker.oci]
  enabled = true
  snapshotter = "auto" # overlayfs or native, default value is "auto".

  gc = true
  # gckeepstorage = 307200

  [[worker.oci.gcpolicy]]
    keepBytes = 8589934592000 #800G
    keepDuration = 172800
    filters = ["type==source.local", "type==exec.cachemount", "type==source.git.checkout"]

  [[worker.oci.gcpolicy]]
     all = true
     keepBytes = 53687091200  # 50G

[registry."hub.antplus.com"]
  mirrors = ["hub.antplus.com"]
  http = true
  insecure = true

[registry."hub.antplus-plus.net"]
  mirrors = ["hub.antplus-plus.net"]
  http = true
  insecure = true
```

启动 buildkitd

方式 一：supervisor

如果服务器上存在 supervisor 程序，在下载压缩包解压的目录下直接执行 `supervisord -c etc/supervisor.conf` 即可。

方式 二：systemd

1. 配置 systemd 的 buildkitd.service，用于启动 buildkitd 程序（systemctl start buildkitd.service）：

```
cat /etc/systemd/system/buildkitd.service

[Unit]
Description=buildkitd
[Service]
User=root
Group=root
Restart=always
PIDFile=/tmp/buildkitd.pid
ExecStart=/opt/local/bin/buildkitd --root=/opt/local/var/lib/buildkit --addr=tcp://0.0.
0.0:23456
ExecReload=/bin/kill -s HUP $MAINPID
```

```
#vim /etc/systemd/system/buildkitd.service

[                         ]
#cat /etc/systemd/system/buildkitd.service
[Unit]
Description=buildkitd
[Service]
User=root
Group=root
Restart=always
PIDFile=/tmp/buildkitd.pid
ExecStart=/opt/local/bin/buildkitd --root=/opt/local/var/lib/buildkit --addr=tcp://0.0.0.0:23456
ExecReload=/bin/kill -s HUP $MAINPID

[                         ]
#systemctl daemon-reload

[                         ]
#systemctl start buildkitd.service
```

> **说明**
>
> 如果 systemctl status buildkitd 查看启动失败，则需要在服务器安装 runc 程序（yum install runc）。

2. 在云游部署参数中调整 buildkitd 的部署地址。

> **说明**
>
> 注意两个参数 BUILDKIT_HOST 与 BUILDKITD_ADDR 之间区别。

分类	键	值模板		描述
常量	ANTB_RUNNER_USER 非空	antb	镜像构建工具 buildkit 运行模式为 C/S，服务端程序是 buildkitd，客户端程序是 builtctl。 此镜像将 runner-flash 与 buildkitd 打至同一个镜像，两个程序启动时监听同一个地址进行访问交互。	ANTB_RUNNER_USER
常量	ANTB_RUNNER_TOKEN 非空 敏感	***	应用 runner-flash 通过调用 buildctl 同 buildkitd 进行交互，默认使用同容器实例中的 buildkitd 服务端。 但是 runner-flash 可以使用远程的 buildkitd 服务端，这时需要修改 BUILDKIT_HOST 参数，改为远程服务的监听地址。	ANTB_RUNNER_TOKEN
常量	BUILDKIT_HOST 非空	unix:///home/admin/local/var/run/buildkitd.sock	应用 runner-flash 需连接的 buildkitd 监听地址	buildkitd connect addr: unix sockets / tcp port
常量	BUILDKITD_ADDR 非空	unix:///home/admin/local/var/run/buildkitd.sock	应用 buildkitd 启动时的监听地址	buildkitd listen addr
表达式引用	ANTB_URL 非空	http://${res.LINKB.lb.antbuild_slb_inner.vip}		ANTB_URL
表达式引用	set_hosts 非空	#if (${prod.LINKB.DEPLOY_TYPE} == 'public') ${prod.LINKB.CODE_INNER_VIP} code.${env.info.domain} #else ${res.ANTCODE.lb.codeinner.vip} code.${env.info.domain} #end		set code hosts
表达式引用	ANTCLOUD_SOFA_storage_type 非空	${prod.LINKB.STORAGE_TYPE}		构建产出物存储类型，aliyun 底座使用 oss，物理机底座使用 s3
表达式引用	hub_ip	#if (${env.prod.HUB.exist}) ${res.HUB.lb.hub_internet_vip.vip} #end		专有云自建 docker hub 的 ip 地址
表达式引用	hub_host	#if (${env.prod.HUB.exist}) http://hub.${env.info.domainname} #end	专有云环境下镜像仓库地址	专有云自建 docker hub 的内部域名

3. 在 Local 站点调整 **BUILDKIT_HOST** 参数。

> **说明**
>
> 注意需要加上 tcp 前缀，如 `tcp://11.166.xx.xx:23456` 。

常量	LINKB	buildrunner	ANTB_RUNNER_USER	antb
常量	LINKB	buildrunner	BUILDKITD_ADDR	unix:///[redacted]td.sock
常量	LINKB	buildrunner	BUILDKIT_HOST	tcp://11.166.[redacted]:23456
表达式引用	LINKB	buildrunner	ANTB_URL	http://${res.[redacted]_slb_inner.vip}
表达式引用	LINKB	buildrunner	ANTCLOUD_SOFA_storage_type	${prod.LINKB.STORAGE_TYPE}

4. 通过云游执行 LINKB 产品的发布部署。

注意事项

由于 antstack plus 底座使用 pvc 作为存储，历史方案中使用宿主机 volumes 的配置会自动挂载成 pvc（会创建成目录），但是 LINKB 的 runner 应用使用了宿主机的文件（非目录）进行挂载会导致部署冲突，如下图所示：

容器 - Volume 配置 (宿主机)　　编辑

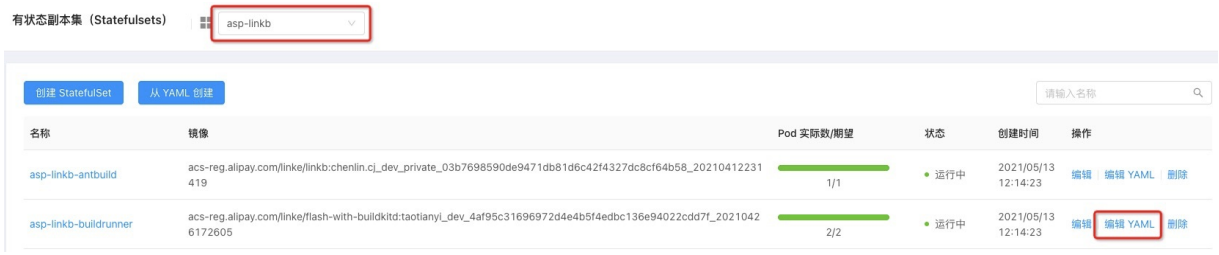
模式	容器路径	宿主机路径	读写模式	Propagation
主机Host	/var/run/docker.sock	/var/run/docker.sock	ro	private
主机Host	/usr/bin/docker	/usr/bin/docker	rw	private
主机Host	/etc/hosts	/etc/hosts	rw	private

21 秒 2021-05-12 17:09:36 ~ 2021-05-12 17:09:57

日志：

```
76b23a3fe1f7373a5162a82dc61f0aa5e700ccda48366e7ea6/rootfs/etc/hosts\\\" caused \\\"not a directory\\\"\"": unknown Reason : StartError].
76b23a3fe1f7373a5162a82dc61f0aa5e700ccda48366e7ea6/rootfs/etc/hosts\\\" caused \\\"not a directory\\\"\"": unknown Reason : StartError].
76b23a3fe1f7373a5162a82dc61f0aa5e700ccda48366e7ea6/rootfs/etc/hosts\\\" caused \\\"not a directory\\\"\"": unknown Reason : StartError].
76b23a3fe1f7373a5162a82dc61f0aa5e700ccda48366e7ea6/rootfs/etc/hosts\\\" caused \\\"not a directory\\\"\"": unknown Reason : StartError].
```

对此需要特殊调整 buildrunner 应用的配置，在 LINKB 产品部署完成初始化后（部署 antbuild 应用时即可操作），进入 captain 找到 buildrunner 的 Statefulsets yaml 配置进行编辑：



名称	镜像	Pod 实际数/期望	状态	创建时间	操作
asp-linkb-antbuild	acs-reg.alipay.com/linke/linkb:chenlin.cj_dev_private_03b7698590de9471db81d6c42f4327dc8cf64b58_20210412231419	1/1	运行中	2021/05/13 12:14:23	编辑 | 编辑 YAML | 删除
asp-linkb-buildrunner	acs-reg.alipay.com/linke/flash-with-buildkitd:taotianyi_dev_4af95c31696972d4e4b5f4edbc136e94022cdd7f_20210426172605	2/2	运行中	2021/05/13 12:14:23	编辑 | 编辑 YAML | 删除

找到 containers 节点下 volumeMounts 配置并进行删除，然后单击 **更新**。

```
      - name: timezone
        value: Asia/Shanghai
      - name: TZ
        value: Asia/Shanghai
      - name: yunqing_container_hostname
        value: asp-linkb-buildrunner-1
      - name: yunqing_container_index
        value: '1'
    resources:
      limits:
        cpu: '2'
        memory: 4Gi
      requests:
        cpu: '2'
        memory: 4Gi
    volumeMounts:
      - name: var-run-docker-sock
        readOnly: true
        mountPath: /var/run/docker.sock
        mountPropagation: None
      - name: usr-bin-docker
        mountPath: /usr/bin/docker
        mountPropagation: None
      - name: etc-hosts
        mountPath: /etc/hosts
        mountPropagation: None
    readinessProbe:
      tcpSocket:
        port: 9252
```

5.4.3. 部署检查

Docker 方案

- 登录 buildrunner 容器，执行 `docker login` 登录 hub 镜像中心，下图表示登录成功。

```
#docker login hub.antstack-plus.net
Username: admin
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

- docker ps 执行正常。

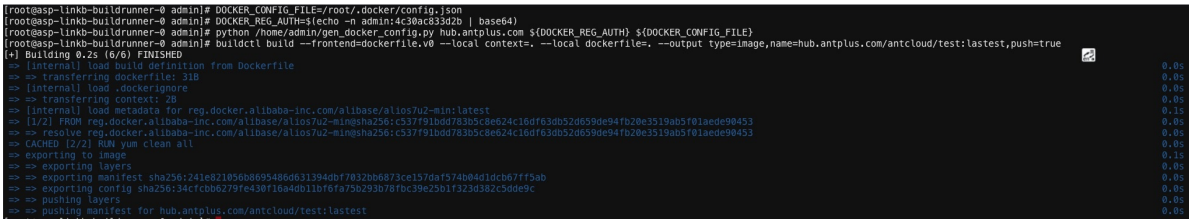
BuildKit 方案

- 登录 buildrunner 容器，用 vim 编写一个简单的 Dockerfile（基础镜像使用 hub 镜像中心能访问到的），登录用户为 `admin` 时，先进行 su root，密码为 `LinkE2017` 。

- 测试一个简单的镜像构建。

> **说明**
>
> `hub.antplus.com` 替换成环境中的 hub 镜像中心域名，`admin:4c30ac833d2b` 替换成 hub 镜像中心的登录账密码。

```
DOCKER_CONFIG_FILE=~/.docker/config.json
DOCKER_REG_AUTH=$(echo -n admin:4c30ac833d2b | base64)
python /home/admin/gen_docker_config.py hub.antplus.com ${DOCKER_REG_AUTH} ${DOCKER_CONFIG_FILE}
buildctl build --frontend=dockerfile.v0 --local context=. --local dockerfile=. --output type=image,name=hub.antplus.com/antcloud/test:lastest,push=true
```



5.5. LINKECI

5.5.1. 部署说明

> **说明**
>
> - 在部署 LINKECI 前，需要完成 LINKEBASE 和 ANT CODE 的部署。
> - 需要确认上一步中 LINKB 使用的部署方案是 Docker 构建还是 BuildKit 构建，然后修改初始化应用 aciinit script 中的 **ACLINKELIB_BUILD_AKS_FRAMEWORK** 参数。

- Docker 构建方案
  - ACLINKELIB_BUILD_AKS_FRAMEWORK：**dockerBuild-v2**。

- ACLINKELIB_BUILD_AKS_FRAMEWORK_VERSION：1.0.1。

- BuildKit 构建方案
  - ACLINKELIB_BUILD_AKS_FRAMEWORK：dockerBuild-v3。
  - ACLINKELIB_BUILD_AKS_FRAMEWORK_VERSION：1.0.0。

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，aciinitscript、linkainitscript、jenkinsregister 的一次性任务（job）必须成功。

5.5.3. 部署检查

暂不需要。

5.6. LINKEPORTAL

5.6.1. 部署说明

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

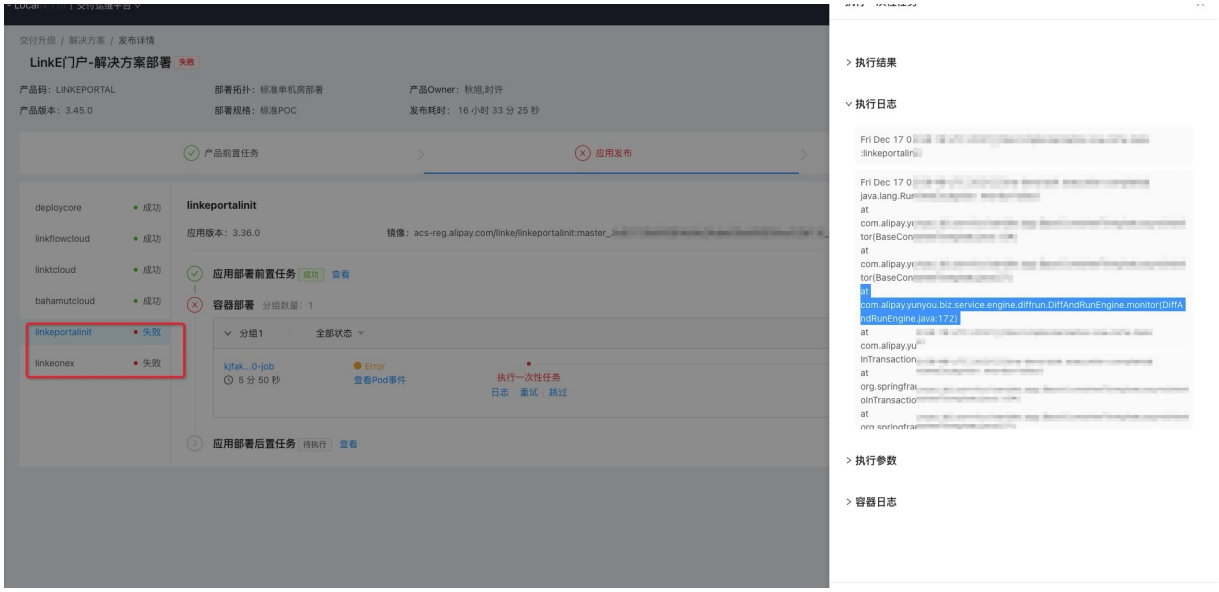
5.6.2. 部署方式

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，linkeportalinit 的一次性任务（job）必须成功。

注意事项

在部署 linkeportalinit 时，提前检查其 mongodb 的部署参数，如下图中的 `aclinkelib_db_url`，在右边渲染出的结果中，`db url` 中存在 3 个一样的 IP 地址，这可能会影响 mongodb 数据的初始化，从而导致部署报错。



修改方案为：在值模板中的3个重复的 mongomaster.vip 中，去除两个仅保留一个：

```
 mongodb://linke:LinkE2017@${res.LINKEBASE.lb.mongomaster.vip}:27018/aclinkelib_main?authSource=admin
```

，如下图所示。

5.6.3. 部署检查

暂不需要。

6.前端页面部署

什么是 OneConsole

LinkE 的前端页面已托管至 Oneconsole 产品，这也是 SOFA 全家产品前端页面的托管地址。

> 注意
>
> 每次部署前，需要找 oneconsole 产品负责人确认其版本，否则若前端页面没有打包正确，其他产品（如 SOFAStack、RMS 等）已有的前端页面会被冲掉覆盖导致页面打不开。

配置 OneConsole

1. 进入 onexadmin 站点，在左侧边栏点击 **控制台**。
2. 找到名字为 **LINKCONSOLE** 的控制台点进去，找到 **原始区块信息** 并点击复制按钮。

3. 单击 **基本信息 > 更新**。

4. 在 **区块信息** 模块，复制第 2 步的配区块信息。

   例如以下参数信息：

```
{
  "displayNameCN": "研发效能平台",
  "icon": "/static-assets/linkconsole/TB1C4G7UNTpK1RjSZFMXXbG_VXa-100-112-7bb63a47.png"
,
  "displayNameEN": "LinkE",
```

```
  "links": [],
  "menus": [
    {
      "docCenter": "",
      "children": [],
      "hasChildren": false,
      "displayNameCN": "总览",
      "icon": "home",
      "link": "#/linke/${tenantName}/dashboard",
      "displayNameEN": "dashboard"
    },
    {
      "docCenter": "",
      "children": [
        {
          "children": [],
          "hasChildren": false,
          "displayNameCN": "我的迭代",
          "link": "#/linke/${tenantName}/iterations",
          "displayNameEN": "iterations",
          "disabled": false
        },
        {
          "children": [],
          "hasChildren": false,
          "displayNameCN": "我的发布",
          "link": "#/linke/${tenantName}/release",
          "displayNameEN": "release",
          "disabled": false
        },
        {
          "children": [],
          "hasChildren": false,
          "displayNameCN": "我的应用",
          "link": "#/linke/${tenantName}/app",
          "displayNameEN": "app",
          "disabled": false
        },
        {
          "children": [],
          "hasChildren": false,
          "displayNameCN": "架构域",
          "link": "#/linke/${tenantName}/archdomain",
          "displayNameEN": "Arch Domain",
          "disabled": false
        }
      ],
      "hasChildren": true,
      "displayNameCN": "研发迭代",
      "icon": "book",
      "link": "",
      "displayNameEN": ""
    },
    {
```

```
    "docCenter": "",
    "children": [],
    "hasChildren": false,
    "displayNameCN": "代码服务",
    "icon": "code",
    "link": "/antcode/dashboard/projects",
    "displayNameEN": "antcode"
  },
  {
    "docCenter": "",
    "children": [
      {
        "children": [],
        "hasChildren": false,
        "displayNameCN": "应用服务",
        "link": "#/linkw/${tenantName}/appService",
        "displayNameEN": "App Instance",
        "disabled": false
      },
      {
        "children": [],
        "hasChildren": false,
        "displayNameCN": "操作记录",
        "link": "#/linkw/${tenantName}/order",
        "displayNameEN": "Operation Log",
        "disabled": false
      }
    ],
    "hasChildren": true,
    "displayNameCN": "研发环境",
    "icon": "cloud",
    "link": "#/linke/${tenantName}/dashboard",
    "displayNameEN": "deploycore"
  },
  {
    "docCenter": "",
    "children": [
      {
        "children": [],
        "hasChildren": false,
        "displayNameCN": "我的项目",
        "link": "#/linkt",
        "displayNameEN": "My Projects",
        "disabled": false
      },
      {
        "children": [],
        "hasChildren": false,
        "displayNameCN": "项目列表",
        "link": "#/linkt/projects/list",
        "displayNameEN": "Project List",
        "disabled": false
      }
    ],
```

```
    "hasChildren": true,
    "displayNameCN": "项目管理",
    "icon": "project",
    "link": "#/linke/${tenantName}/dashboard",
    "displayNameEN": "linkt"
  },
  {
    "docCenter": "",
    "children": [],
    "hasChildren": false,
    "displayNameCN": "定时任务",
    "icon": "code",
    "link": "#/linke?navPath=ProjectCI",
    "displayNameEN": "Code Integration"
  },
  {
    "displayNameCN": "流程中心",
    "displayNameEN": "",
    "icon": "check-square",
    "docCenter": "",
    "hasChildren": true,
    "link": "#/linkflow",
    "children": [
      {
        "displayNameCN": "LinkE 流程(LinkFlow)",
        "displayNameEN": "LinkFlow",
        "hasChildren": false,
        "link": "#/linkflow",
        "disabled": false,
        "children": []
      },
      {
        "displayNameCN": "三方审批流程",
        "displayNameEN": "",
        "hasChildren": false,
        "link": "#/linke/${tenantName}/bahamut/tripartite",
        "disabled": false,
        "children": []
      }
    ]
  },
  {
    "docCenter": "",
    "children": [
      {
        "children": [],
        "hasChildren": false,
        "displayNameCN": "部署配置",
        "link": "#/linke/${tenantName}/admin/workspace",
        "displayNameEN": "Deploy Config",
        "disabled": false
      },
      {
        "children": [],
        "hasChildren": false,
```

```
          "hasChildren": false,
          "displayNameCN": "流水线配置",
          "link": "#/linke/${tenantName}/admin/pipeline",
          "displayNameEN": "Pipeline Config",
          "disabled": false
        },
        {
          "displayNameCN": "审批配置",
          "displayNameEN": "",
          "hasChildren": false,
          "link": "#/linke/${tenantName}/admin/process",
          "disabled": false,
          "children": []
        },
        {
          "children": [],
          "hasChildren": false,
          "displayNameCN": "迭代配置",
          "link": "#/linke/${tenantName}/admin/template",
          "displayNameEN": "Iteration Config",
          "disabled": false
        },
        {
          "children": [],
          "hasChildren": false,
          "displayNameCN": "构建配置",
          "link": "#/linke/${tenantName}/admin/build",
          "displayNameEN": "Build Config",
          "disabled": false
        },
        {
          "children": [],
          "hasChildren": false,
          "displayNameCN": "角色配置",
          "link": "#/linke/${tenantName}/admin/userRole",
          "displayNameEN": "Role Config",
          "disabled": false
        }
      ],
      "hasChildren": true,
      "displayNameCN": "管理员配置",
      "icon": "setting",
      "link": "#/linke?navPath=dashboard",
      "displayNameEN": "Administrator Settings"
    }
  ],
  "type": "platform",
  "actions": [],
  "apps": [
    {
      "routerPrefix": "#/linke",
      "apiDomain": "http://linke.{domain_name}",
      "name": "bahamut",
      "entryHTML": "/static-assets/linkconsole/bahamut/index-d987a5d3.html"
    },
```

```
    },
    {
      "routerPrefix": "#/linkw",
      "apiDomain": "http://deploycore.{domain_name}",
      "name": "deploycore",
      "entryHTML": "/static-assets/linkconsole/deploycore/index-8ed4b303.html"
    },
    {
      "routerPrefix": "#/linkt",
      "apiDomain": "http://linkt.{domain_name}",
      "name": "linkt",
      "entryHTML": "/static-assets/linkconsole/linkt/index-3c506a55.html"
    },
    {
      "routerPrefix": "#/linkflow",
      "apiDomain": "http://linkflow.{domain_name}",
      "name": "linkflow",
      "entryHTML": "/static-assets/linkconsole/linkflow/index-5dc7aee6.html"
    },
    {
      "name":"antcode",
      "entryHTML":"http://code.kf.zjnx.net/index.html",
      "routerPrefix":"/antcode",
      "apiDomain":"http://code.{domain_name}"
    },
    {
      "name": "linkq",
      "entryHTML": "http://linkq.kf.zjnx.net/tc/index",
      "routerPrefix": "/tc",
      "apiDomain": "http://linkq.{domain_name}"
    },
    {
      "routerPrefix": "#/linka",
      "apiDomain": "http://linka.{domain_name}",
      "name": "linka",
      "entryHTML": "/static-assets/linkconsole/linka/index-5e16a5da.html"
    },
    {
      "routerPrefix": "#/fabric",
      "apiDomain": "http://linke.{domain_name}/fabric",
      "name": "fabric",
      "entryHTML": "/static-assets/linkconsole/fabric/index-3957c138.html"
    },
    {
      "routerPrefix": "#/devmind",
      "apiDomain": "http://devmind.{domain_name}",
      "name": "devmind",
      "entryHTML": "/static-assets/linkconsole/devmind/index-0e9ab5d7.html"
    }
  ]
}
```

找到 `AntCode` 相关的配置项并把 `entryHTML` 改为 `http://code.{环境域名}/index.html`，在实际部署中，请使用客户实例域名进行替换。

5. 单击 **更新**，完成前端部署。

7.LinkE 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。

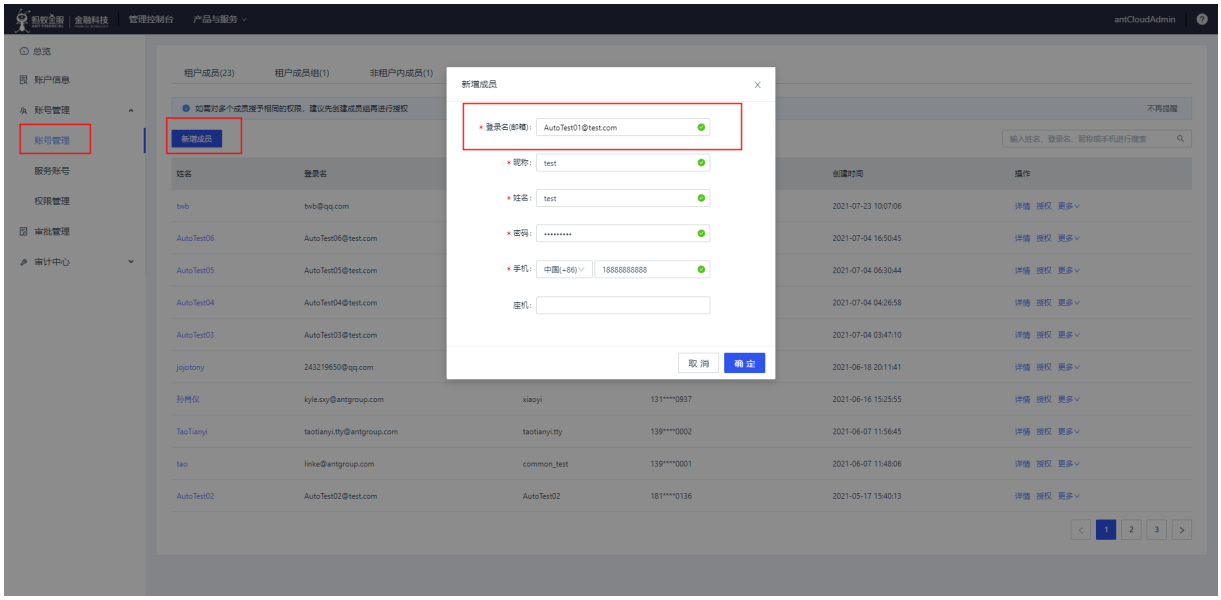
自动化测试配置有 3 个方面：初始化用户、LINKEPORTAL、ANTCODE，下文分别介绍相应的操作步骤。

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

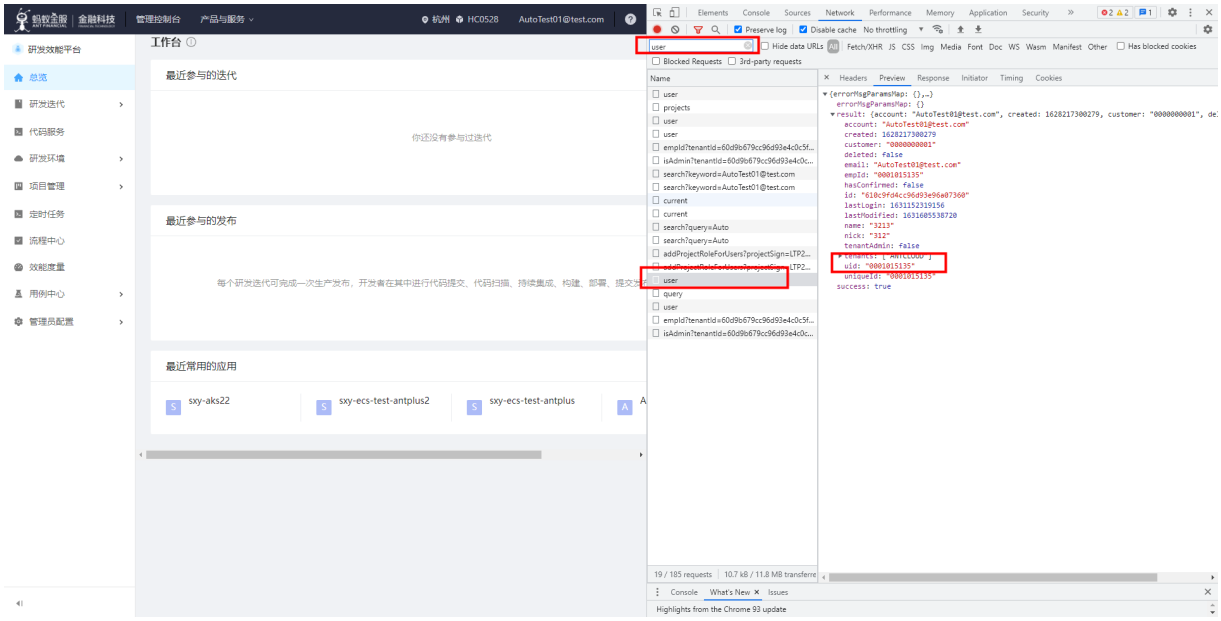
注册完成后，使用以下 3 个账号分别登录 LinkE。

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

> **注意**
>
> 此步骤必须进行，否则自动化配置无效！

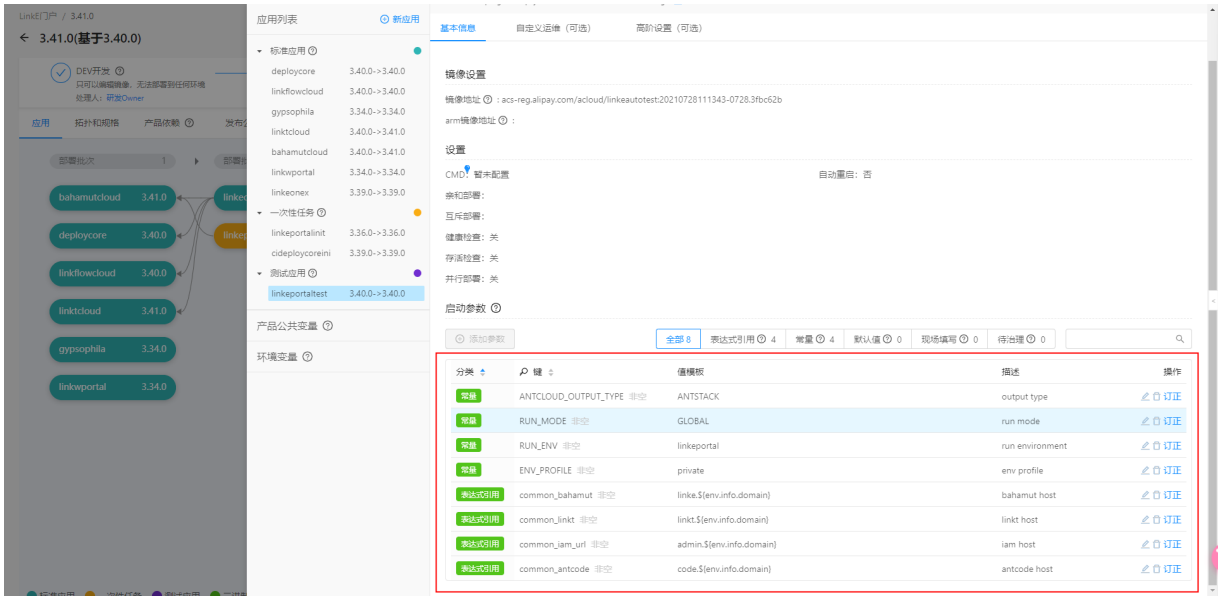
步骤三：获取 AutoTest01@test.com 账号 ID

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。

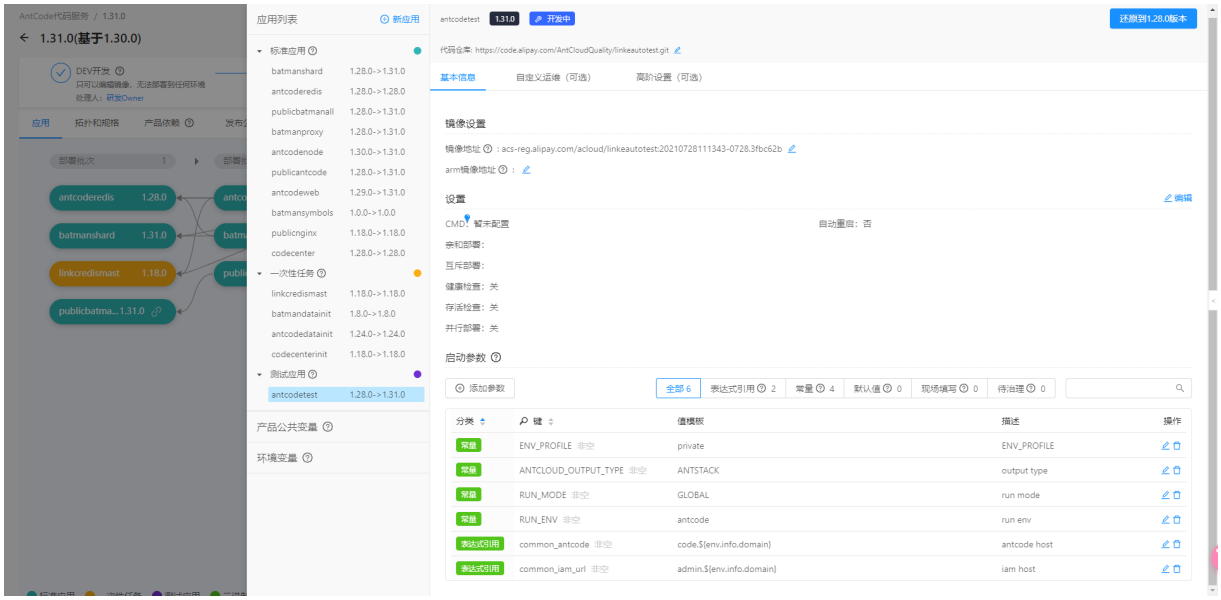


键	值模板	描述	是否现场填写
ANTCLOUD_OUTPUT_TYPE	ANTSTACK	output type	否

键	值模板	描述	是否现场填写
RUN_MODE	GLOBAL	run mode	否
RUN_ENV	linkeportal	run environment	否
ENV_PROFILE	private	env profile	否
common_bahamut	linke.${env.info.domain}	bahamut host	否
common_linkt	linkt.${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	linkflow.${env.info.domain}	linkflow host	否
common_linke	linkconsole.${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 获取方法请参见 步骤三。
common_schema	http:// 或者 https://	http 协议头	否

ANTCODE

登录云游平台， 进入 **ANTCODE** 应用页面，查看/修改测试应用 **antcodetest** 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
ENV_PROFILE	private	ENV_PROFILE	否
ANTCLOUD_OUTPUT_TYPE	ANTSTACK	output type	否
RUN_MODE	GLOBAL	run mode	否
RUN_ENV	antcode	run env	否
common_antcode	code.${env.info.domain}	antcode host	否
common_iam_url	admin.${env.info.domain}	iam host	否
common_linkflow	linkflow.${env.info.domain}	linkflow host	是
common_user2ID		在初始化用户中，AutoTest01@test.com 账号 ID。	是 获取方法请参见 步骤三。

键	值模板	描述	是否现场填写
common_user2	AutoTest02@test.com	在初始化用户中， AutoTest02@test.com 账号 ID。	是 获取方法请参见 步骤一。

8.产品访问

8.1. 负载均衡

部署完成后访问需要先配置 LinkE 产品的对应域名才能正常访问控制台。

阿里云底座

SLB 自动创建并设置监听，支持 TCP 四层协议和 HTTP 七层协议，域名需要单独配置，解析到对应 SLB IP。

AntStackPlus 底座

ALB 仅支持七层协议，不支持四层 TCP 协议，需要支持一个域名解析到多个 IP，或者其他负载均衡系统如 F5 手动配置高可用能力，需要检查所有名称为 `link*/code*/easycase` 的负载均衡实例，如健康检查失败导致状态异常需先关闭 ALB 的健康检查。

8.2. 访问域名

请参见 负载均衡和域名，查询对应应用的 IP 配置 hosts。

9.常见问题

9.1. 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;
```

9.2. 服务启动时报 ZoneClientUtil 初始化错误

问题现象

服务启动时报 ZoneClientUtil 初始化错误。

问题原因

ZoneClientUtil 初始化失败。

解决方案

ZoneClientUtil 是 SOFA CE 框架自带的单元化配置工具，LinkE 去除了单元化配置依赖，如果其初始化失败，需要将启动参数中 linku、metacenter、fabric 等相关应用部署参数中的 `cell` 与 `datacenter` 的值设置为 `defalut` 。

后续解决方案

- 将去除单元化配置项写入 `sofa-config.properties` 配置中。

```
# middleware
com.alipay.env=${com_alipay_env}
com.alipay.instanceid=${com_alipay_instanceid}
com.antcloud.antvip.endpoint=${com_antcloud_antvip_endpoint}

#datacenter=${datacenter}
#cell=${cell}

# disabled zmode: LDC, 取消对单元化环境的依赖
zmode=false
datacenter=default
cell=default

# disable log4j
# https://yuque.antfin-inc.com/docs/share/1db80a1a-f420-4f76-8
lookout_enable=false
```

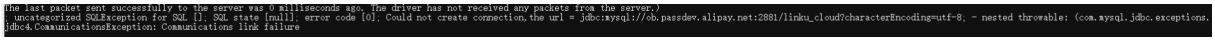
> 说明
>
> 该配置项已于 2022 年 1 月 完成，已经合入应用镜像中。

- 将相关应用技术栈升级为 SOFABOOT。

9.3. Zdal 框架连接数据库失败

问题现象

如下图所示，服务启动时报数据库连接不上的错误：



问题原因

手动可以连接数据库，但是启动时数据库连接失败，经排查，是由于 LinkE 某些应用（如 Linku）使用的 Zdal 框架不支持非 3306 默认端口的 MySQL。

解决方案

将 MySQL 默认端口修改为 3306。

> **说明**
>
> - 已经尝试修改过
>
> `/home/admin/release/run/linku.ace/config/db/linku_1.0_zdal-linku/groupCluster.zdal-linku.json`
>
> 中的配置，经验证数据库端口配置不生效，当前此版本 zdal 框架仅支持 3306 端口 mysql。
> - 问题 zdal 版本为：zdal-client-5.3.0

后续处理方案

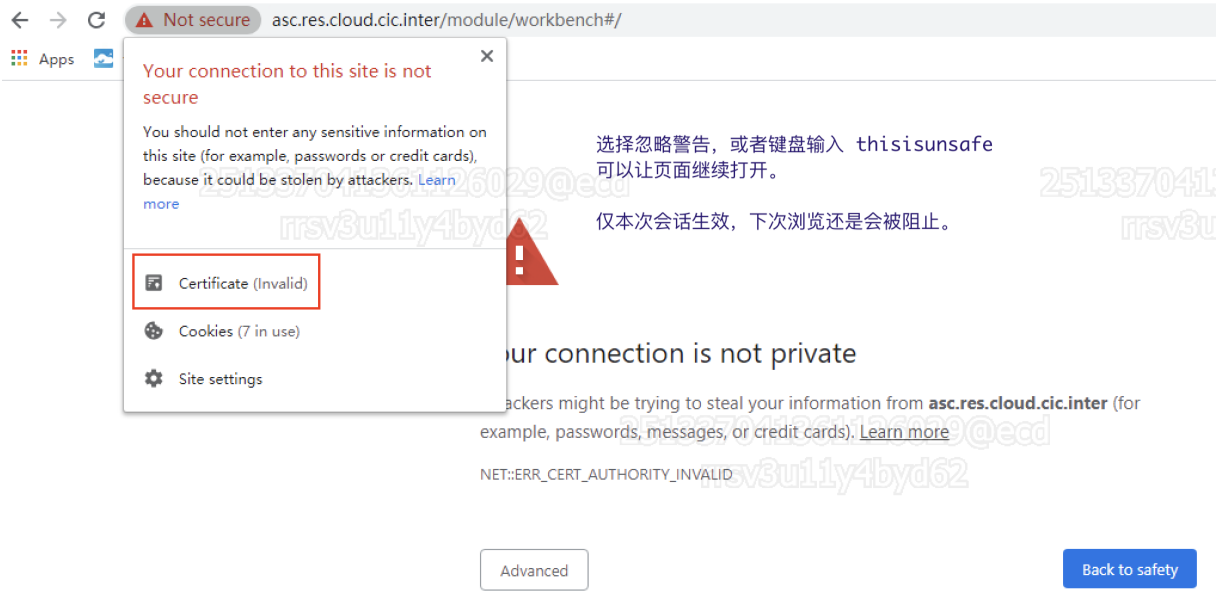
- 寻找合适的版本与配置，让 Zdal 支持非默认 3306 端口的 MySQL。
- 去除 Zdal 依赖，使用原生的 spring jdbc 或 alibaba druid 连接池。

9.4. 镜像构建支持自签证书镜像仓库部署方案

镜像仓库自签证书是什么

在客户的专有云环境中，域名通常不是公开切仅内部使用。在这种场景下，如果客户有添加 HTTPS 访问支持的需求，通常会采取自签证书方案。

如下图所示，使用 Chrome 打开页面提示为非安全页面，提示证书为 **Invalid**。



> **说明**
>
> 选择忽略警告，或者键盘输出 thisisunsafe 可以让页面继续打开。仅本次会话生效，下次浏览还是会被阻止。

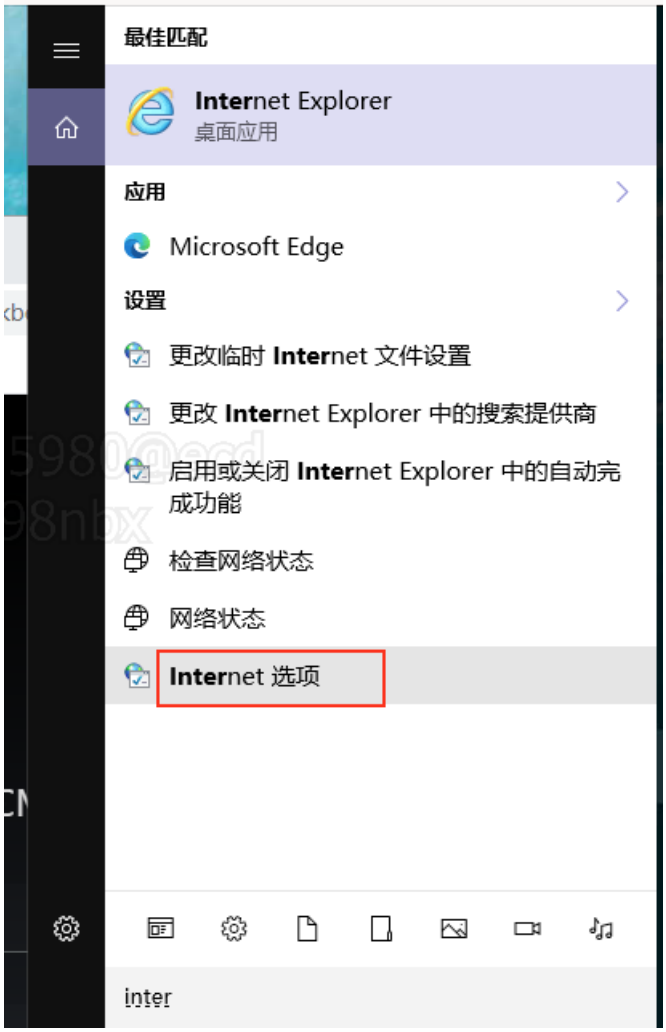
自签证书使用方式

浏览器级别使用方式

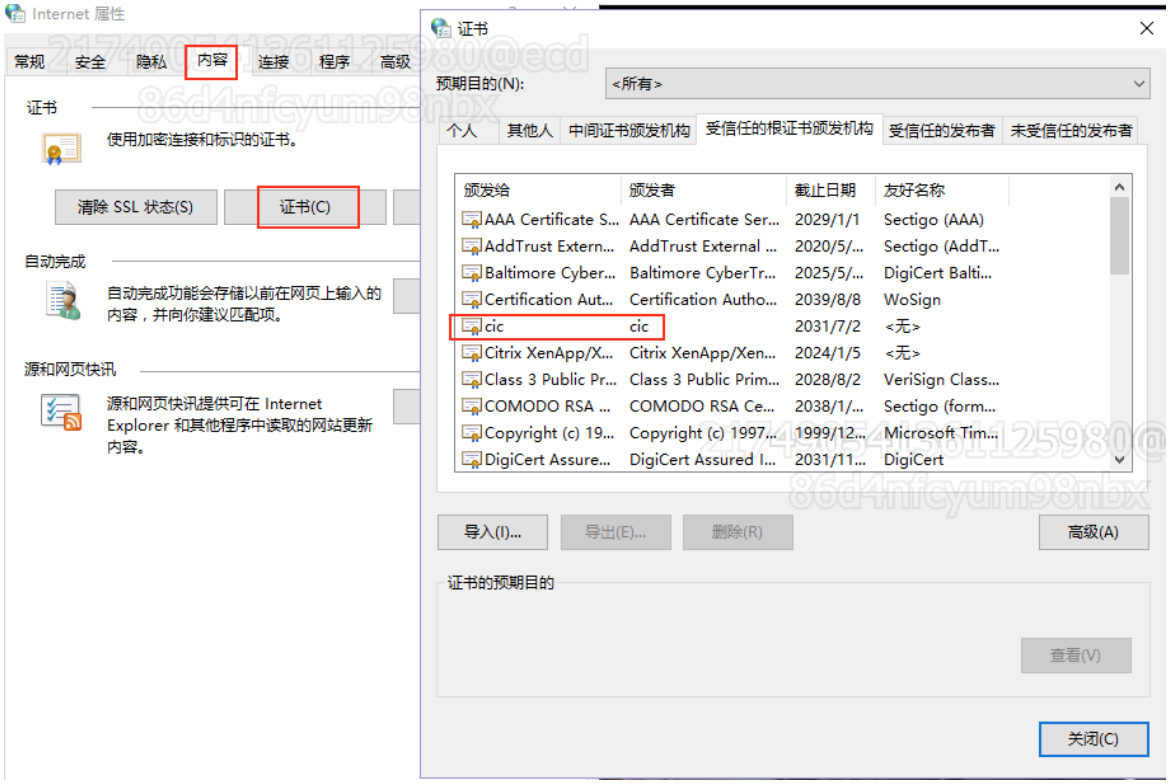
以 Windows 系统为例，操作步骤如下：

1. 在 Chrome 浏览器中，导出证书。

2. 打开 Internet 选项。



3. 在受信任的根证书区域中导入证书。



4. 关闭浏览器重新打开生效。

Linux 服务器使用方式

由 Windows 系统的操作步骤可知，根证书会直接安装在系统文件目录中。如果操作系统是 Linux，如下图所示，会表现为：**使用 curl 命令提示证书不受信任，拒绝连接。**

```
[root@zhcxtest-linkeci-aclinkelibcloud-0 admin]# curl -v https://cr.authentication.res.cloud.cic.inter
* About to connect() to cr.authentication.res.cloud.cic.inter port 443 (#0)
*   Trying 10.206.41.179...
* Connected to cr.authentication.res.cloud.cic.inter (10.206.41.179) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
*   CAfile: /etc/pki/tls/certs/ca-bundle.crt
  CApath: none
* Server certificate:
*       subject: CN=*.res.cloud.cic.inter
*       start date: 7月 01 16:00:00 2021 GMT
*       expire date: 7月 01 16:00:00 2031 GMT
*       common name: *.res.cloud.cic.inter
*       issuer: CN=cic
* NSS error -8172 (SEC_ERROR_UNTRUSTED_ISSUER)
* Peer's certificate issuer has been marked as not trusted by the user.
* Closing connection 0
curl: (60) Peer's certificate issuer has been marked as not trusted by the user.
More details here: http://curl.haxx.se/docs/sslcerts.html

curl performs SSL certificate verification by default, using a "bundle"
 of Certificate Authority (CA) public keys (CA certs). If the default
 bundle file isn't adequate, you can specify an alternate file
 using the --cacert option.
If this HTTPS server uses a certificate signed by a CA represented in
 the bundle, the certificate verification probably failed due to a
 problem with the certificate (it might be expired, or the name might
 not match the domain name in the URL).
If you'd like to turn off curl's verification of the certificate, use
 the -k (or --insecure) option.
[root@zhcxtest-linkeci-aclinkelibcloud-0 admin]#
```

此种情况有以下两种解决方案：

- **忽略证书**

  使用 curl 时增加 `-k` 参数，如下图所示。

```
> curl -k -v https://cr.authentication.res.cloud.cic.inter
* About to connect() to cr.authentication.res.cloud.cic.inter port 443 (#0)
*   Trying 10.206.41.179...
* Connected to cr.authentication.res.cloud.cic.inter (10.206.41.179) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* skipping SSL peer certificate verification
* SSL connection using TLS ECDHE RSA WITH AES 128 GCM SHA256
* Server certificate:
*       subject: CN=*.res.cloud.cic.inter
*       start date: 7月 01 16:00:00 2021 GMT
*       expire date: 7月 01 16:00:00 2031 GMT
*       common name: *.res.cloud.cic.inter
*       issuer: CN=cic
> GET / HTTP/1.1
> User-Agent: curl/7.29.0
> Host: cr.authentication.res.cloud.cic.inter
> Accept: */*
>
< HTTP/1.1 404 Not Found
< Server: Tengine
< Date: Wed, 19 Jan 2022 08:12:59 GMT
< Content-Length: 0
< Connection: close
< Set-Cookie: XSRF-TOKEN=80871f70-94ed-4bb8-a854-7869c597e8ed; Path=/; HttpOnly
< X-Content-Type-Options: nosniff
< X-XSS-Protection: 1; mode=block
< Cache-Control: no-cache, no-store, max-age=0, must-revalidate
< Pragma: no-cache
< Expires: 0
< X-Frame-Options: DENY
<
* Closing connection 0
```

> **说明**
>
> 这种忽略证书方案需要在编码时就植入，一般在自测时使用。

- **系统植入根证书**

  需要使用 openssl 工具，服务器默认已安装：

  > **说明**
  >
  > https://daniel.haxx.se/blog/2018/11/07/get-the-ca-cert-for-curl/

  假设客户自签证书网站为 `https://cr.registry.res.cloud.cic.inter`，操作步骤如下：

  i. 命令行使用 openssl 下载证书。

```
server=cr.registry.res.cloud.cic.inter

# 下载证书
echo quit | openssl s_client -showcerts -servername $server -connect $server:443 > /t
mp/cacert.pem

# 验证证书是否有效
curl -Iv --cacert /tmp/cacert.pem https://$server

# 更新系统根证书
cp -p /tmp/cacert.pem /etc/pki/ca-trust/source/anchors
update-ca-trust

# 验证系统根证书是否已更新
curl -Iv https://$server
```

ii. 借助 windows / mac 使用浏览器下载后手动上传。

```
# 2. 可以通过 windows / mac 使用浏览器下载后手动上传
# /tmp/cic.cer 为上一步在 windows 导出的 X509 格式 DER 编码类型证书

# 查看证书
openssl x509 -inform der -in /tmp/cic.cer -noout -text

# 转换证书
cd /etc/pki/ca-trust/source
openssl x509 -inform der -in /tmp/cic.cer -out anchors/cic.crt

# 更新根证书
update-ca-trust

# 验证系统根证书是否已更新
server=cr.registry.res.cloud.cic.inter
curl -Iv https://$server
```

其中 `update-ca-trust` 是一个脚本，如果提示无此命令，可自行创建一个：

```
[root@zhcxtest-linkeci-aclinkelibcloud-0 admin]# which update-ca-trust
/usr/bin/update-ca-trust
[root@zhcxtest-linkeci-aclinkelibcloud-0 admin]# cat /usr/bin/update-ca-trust
#!/bin/sh

#set -vx

# At this time, while this script is trivial, we ignore any parameters given.
# However, for backwards compatibility reasons, future versions of this script mu
st
# support the syntax "update-ca-trust extract" trigger the generation of output
# files in $DEST.

DEST=/etc/pki/ca-trust/extracted

# OpenSSL PEM bundle that includes trust flags
# (BEGIN TRUSTED CERTIFICATE)
/usr/bin/p11-kit extract --comment --format=openssl-bundle --filter=certificates
--overwrite $DEST/openssl/ca-bundle.trust.crt
/usr/bin/p11-kit extract --comment --format=pem-bundle --filter=ca-anchors --over
write --purpose server-auth $DEST/pem/tls-ca-bundle.pem
/usr/bin/p11-kit extract --comment --format=pem-bundle --filter=ca-anchors --over
write --purpose email $DEST/pem/email-ca-bundle.pem
/usr/bin/p11-kit extract --comment --format=pem-bundle --filter=ca-anchors --over
write --purpose code-signing $DEST/pem/objsign-ca-bundle.pem
/usr/bin/p11-kit extract --format=java-cacerts --filter=ca-anchors --overwrite --
purpose server-auth $DEST/java/cacerts
```

重新运行 curl 命令，可以看到访问已经受信任：

```
> curl -v https://cr.authentication.res.cloud.cic.inter
* About to connect() to cr.authentication.res.cloud.cic.inter port 443 (#0)
*   Trying 10.206.41.179...
* Connected to cr.authentication.res.cloud.cic.inter (10.206.41.179) port 443 (#0
)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
*   CAfile: /etc/pki/tls/certs/ca-bundle.crt
  CApath: none
* SSL connection using TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
* Server certificate:
*       subject: CN=*.res.cloud.cic.inter
*       start date: 7月 01 16:00:00 2021 GMT
*       expire date: 7月 01 16:00:00 2031 GMT
*       common name: *.res.cloud.cic.inter
*       issuer: CN=cic
> GET / HTTP/1.1
> User-Agent: curl/7.29.0
> Host: cr.authentication.res.cloud.cic.inter
> Accept: */*
>
< HTTP/1.1 404 Not Found
< Server: Tengine
< Date: Wed, 19 Jan 2022 08:19:01 GMT
< Content-Length: 0
< Connection: close
< Set-Cookie: XSRF-TOKEN=d7b676de-1320-434c-bac2-48557c7b80d8; Path=/; HttpOnly
< X-Content-Type-Options: nosniff
< X-XSS-Protection: 1; mode=block
< Cache-Control: no-cache, no-store, max-age=0, must-revalidate
< Pragma: no-cache
< Expires: 0
< X-Frame-Options: DENY
<
* Closing connection 0
```

镜像构建工具 buildrunner 对自签证书的支持

从 2.25 LINKB 产品集开始，部署新增 **CERT_URLS** 参数，默认为空，可填写空值。

如果客户有诉求使用自签名镜像仓库，需要提前按上述步骤（推荐使用 openssl 下载）准备好私签证书，存储于客户现场某个 HTTP 服务器（推荐使用客户现场可公开权限访问的 OSS Bucket），复制下载链接填入。

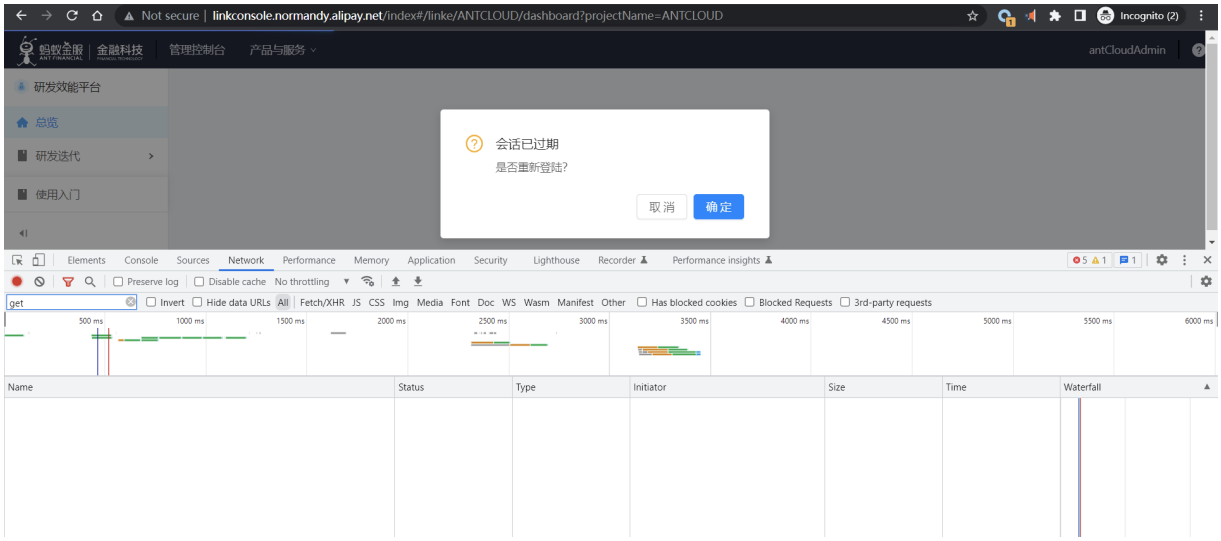
> **说明**
>
> 如果有多个私签证书，可使用逗号,分割，如可填写：
> http://oss.custom.com/cert1.pem,http://oss.custom.com/cert2.pem

应用 buildrunner 在启动时会尝试下载私签证书，并写入 POD 的系统根证书区域，从而杜绝各种因私签证书导致的构建镜像时出现拉不下来推不上去问题。

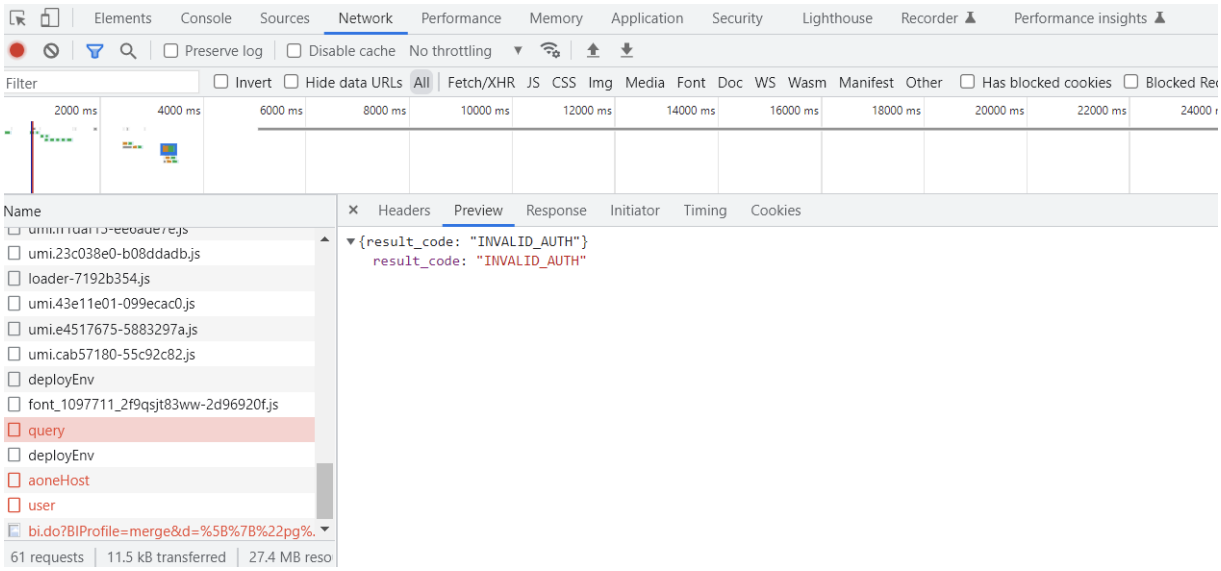
9.5. LinkE 产品卸载再重复部署后账号无法登录

问题现象

部署完成后，进入 LinkE 首页，一直提示没有登录。



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



问题原因

LinkE 在部署后，会在 IAM ROC 租户下，创建一个 LINKE专用服务账号。

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。

如果 LinkE 产品被卸载，需要手动在 IAM 上删除此服务号。否则再次部署时，IAM 服务号因同号将创建不成功，阻塞 LinkE 产品的用户登录态查询请求。

如果产品卸载后没有删除此服务账号，又重新部署了 LinkE 产品，则此时需要订正 linku 应用的数据库。

解决方法

(1) LinkE 专用服务号的信息查询

1. IAM 控制台切换到 ROC 租户：

2. 查看 LinkE 服务号信息：

(2) LinkE 应用 linku 数据库已保存信息的查询

命令行登录 linkebase 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。

```
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
```

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。

> **说明**
>
> 如下图所示，其 ip 地址为 172.16.224.184，则 op endpoint 为 http://172.16.224.184/gateway.do。

(3) 手动订库

其中 ak/sk 需要在 IAM 控制台上获取。

```
-- 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 ...
```