zPaaS低代码开发平台部署指南

接上一篇《zPaaS低代码平台使用介绍：实现AI多轮对话功能》，本篇主要介绍一下如何部署私有的zPaaS低代码开发平台。

1. 部署需求

（1）一台linux服务器，配置要求不高，建议2核4G以上的配置，对存储没有特殊要求
（2）已经安装jdk21
（3）一个mysql实例
（4）两套ssl证书，分别用于前后端的https部署

2. 数据库配置

（1）创建低代码平台主数据库zpaas_lowcode_revise

```
create database  zpaas_lowcode_revise;
grant all on zpaas_lowcode_revise.* to linux@'%';
alter user linux@'%' identified with mysql_native_password BY 'linux';
flush privileges;
```

（2）在zpaas_lowcode_revise库中执行《zpaas_lowcode_revise-ddl.sql》文件中的建表语句

（3）导入《zpaas-lowcode-sys-conf-data.sql》、《zpaas-lowcode-sm-business-system-data.sql》、《zpaas-lowcode-sm-data.sql》三个文件中的数据到zpaas_lowcode_revise库中

```
mysql -ulinux -p zpaas_lowcode_revise < ./zpaas-lowcode-sys-conf-data.sql
mysql -ulinux -p zpaas_lowcode_revise < ./zpaas-lowcode-sm-data.sql
mysql -ulinux -p zpaas_lowcode_revise < ./zpaas-lowcode-sm-business-system-data.sql
```

（4）创建低代码平台系统管理模块的数据sm

```
create database  sm;
grant all on sm.* to linux@'%';
flush privileges;
```

（5）在sm库中执行《sm-ddl.sql》文件中的建表语句

（6）导入《database-sm.sql》文件中的数据到sm库中

```
mysql  -ulinux -p sm  < ./database-sm.sql
```

（7）如果需要用到工作流，需要创建工作流引擎对应的数据库camunda

```
create database  camunda;
grant all on camunda.* to linux@'%';
flush privileges;
```

3. 后端服务部署

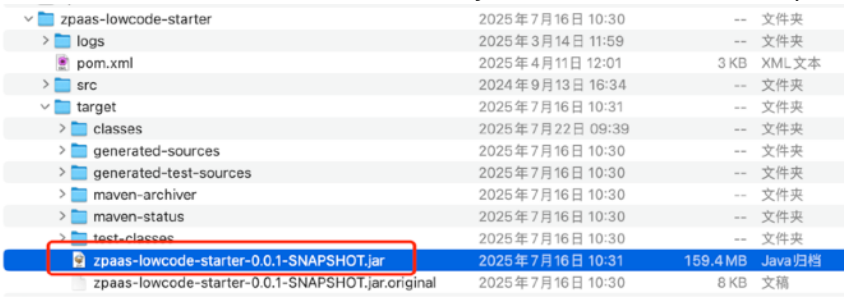
（1）从GitHub仓库https://github.com/zjyzju/zPaaS-lowcode中下载源码，源码的目录结构如下：

zpaas-documents：放置文档及部署脚本
zpaas-lowcode：后端服务工程，其中zpaas-lowcode-starter是启动器模块
zpaas-lowcode-frontend：前端服务工程

（2）编译打包

```
cd zpaas-lowcode
mvn install
cd zpaas-lowcode-starter
mvn package spring-boot:repackage
```

（3）在zpaas-lowcode-starter模块下会打包生成低代码平台后端服务的部署包zpaas-lowcode-starter-0.0.1-SNAPSHOT.jar，该部署包是典型的SpringBoot启动一体包。



（4）将部署包zpaas-lowcode-starter-0.0.1-SNAPSHOT.jar上传到服务器，该部署包的目录结构如下：

application.yml：配置文件

lowcode.keystore：https证书

（5）使用预先准备的https证书替换部署包中的证书

```
jar -uvf zpaas-lowcode-starter-0.0.1-SNAPSHOT.jar BOOT-INF/classes/lowcode.keystore
```

（6）使用预告准备的application.yml配置文件替换部署包中的配置文件

```
jar -uvf zpaas-lowcode-starter-0.0.1-SNAPSHOT.jar BOOT-INF/classes/application.yml
```

（7）application.yml文件说明，可以使用以下命令从部署包中获取

```
jar -xvf zpaas-lowcode-starter-0.0.1-SNAPSHOT.jar BOOT-INF/classes/application.yml
```

```
server:
    port: 18043    #后端服务绑定的端口
    address: 0.0.0.0        #后端服务绑定的ip地址
    ssl:                #HTTPS证书配置
        protocol: TLS
        key-store: classpath:lowcode.keystore
        key-store-password: ENC(xxxx)
        key-store-type: JKS
    servlet:
        context-path: /lowcode        #后端服务的上下文配置
        session:
            timeout: 7200    #Session失效时间配置，单位为秒

spring:
    application:
        name: lowcode-project        #Spring应用的名称

    datasource:
        lcdp-engine:            #低代码平台主数据库配置，用户名密码需要进行相应的调整
            username: linux
            password: ENC(xxxx)
            jdbcUrl: jdbc:mysql://10.0.0.6:3306/zpaas_lowcode_revise?
serverTimezone=UTC&useUnicode=true&characterEncoding=utf-8
            driver-class-name: com.mysql.cj.jdbc.Driver
        camunda:                #工作流引擎数据库配置，用户名密码需要进行相应的调整
            driver-class-name: com.mysql.cj.jdbc.Driver
            jdbcUrl: jdbc:mysql://10.0.0.6:3306/camunda?
serverTimezone=UTC&useUnicode=true&characterEncoding=utf-8
            username: linux
            password: ENC(xxxx)

    servlet:
        multipart:                #上传文件的大小限制
            enabled: true
            max-file-size: 10MB
            max-request-size: 20MB

#jasypt加密,将密钥作为环境变量参数在执行应用的启动命令时传入
jasypt:
    encryptor:                #配置文件中关键信息加密设置
        password: lowcode@ajfiwUU&
        algorithm: PBEWITHHMACSHA512ANDAES_256
```

```
caffeine:                    #业务流编排中本地缓存服务节点使用的配置
    name: caffeineCache
    spec: maximumSize=10000

camunda:          #工作流引擎用到的配置
    bpm:
        admin-user:
            id: demo
            password: ENC(xxxx)
            firstName: Demo
        generic-properties:
            properties:
                historyTimeToLive: P365D
    filter:
        create: All tasks


logging:              #日志输出配置
    level:
        root : info
        cn.zpaas.lowcode : DEBUG
       org.springframework.jdbc: ERROR
        org.apache.kafka.clients: WARN
    pattern:
        console: "%clr(%d{yyyy-MM-dd HH:mm:ss.SSS}){faint} %clr(${LOG_LEVEL_PATTERN:-%5p})
%clr(%-40.40logger{39}){cyan}[%line] %clr(:){faint} %.-1024m%n$
{LOG_EXCEPTION_CONVERSION_WORD:%wEx}"
        #file: "%clr(%d{yyyy-MM-dd HH:mm:ss.SSS}){faint} %clr(${LOG_LEVEL_PATTERN:-%5p})
%clr(%-40.40logger{39}){cyan}[%line] %clr(:){faint} %m%n${LOG_EXCEPTION_CONVERSION_WORD:
%wEx}"
    file:
        name: "zpaas-lowcode.log"
        max-size: 10MB

snowFlake:          #雪花ID生成配置
    dataCenterId: 2
    machineId: 2

### 调度中心部署根地址 [选填]：如调度中心集群部署存在多个地址则用逗号分隔。执行器将会使用该地址进行
"执行器心跳注册"和"任务结果回调"；为空则关闭自动注册；
xxl:
    job:
        admin:
            addresses: http://10.0.0.6:8099/xxl-job-admin
### 执行器通讯TOKEN [选填]：非空时启用；
        accessToken: default_token
### 执行器AppName [选填]：执行器心跳注册分组依据；为空则关闭自动注册
        executor:
            appname: xxl-job-executor-lowcode
### 执行器注册 [选填]：优先使用该配置作为注册地址，为空时使用内嵌服务 ”IP:PORT“ 作为注册地址。从而
更灵活的支持容器类型执行器动态IP和动态映射端口问题。
            address:
### 执行器IP [选填]：默认为空表示自动获取IP，多网卡时可手动设置指定IP，该IP不会绑定Host仅作为通讯实
用；地址信息用于 "执行器注册" 和 "调度中心请求并触发任务"；
            ip:
```

```
### 执行器端口号 [选填]：小于等于0则自动获取；默认端口为9999，单机部署多个执行器时，注意要配置不同执行器端口；
        port: 9999
### 执行器运行日志文件存储磁盘路径 [选填] ： 需要对该路径拥有读写权限；为空则使用默认路径；
        logpath: /Users/zjy/vs_workspace/zpaas-lowcode/zpaas-lowcode-starter/logs
### 执行器日志文件保存天数 [选填] ： 过期日志自动清理, 限制值大于等于3时生效; 否则, 如-1, 关闭自动清理功能；
        logretentiondays: 30

init:               #基于低代码开发的后端服务提供服务时的配置参数
  param:
    systemId:           #单业务系统部署时配置
    tenantId:           #单业务系统部署时配置
    servicePath: service                                    #常规服务请求上下文
    streamServicePath: streamService                        #流式服务请求上下文
    batchServicePath: batchService                          #批量服务请求上下文
    fileUploadServicePath: fileUploadService                #文件上传服务请求上下文
    fileDownloadServicePath: fileDownloadService            #文件下载服务请求上下文
    tempFilePath: /opt/zpaas-lowcode/tempFile/              #文件处理时的临时目录
    loginInfoKey: loginInfo                                 #默认的session中登录信息存储的key
    stream:
      timeout: 60000                                        #流式服务请求的超时时间，单位ms

service:            #多实例部署时，相关的配置
  registry:
    registryTime: 60000
    registryTimeout: 120000
```

（8）启动命令

```
nohup java -jar -server /opt/zpaas-lowcode/zpaas-lowcode-starter-0.0.1-SNAPSHOT.jar &
```

（9）如果需要使用docker方式部署，可以参考以下脚本

制作基础镜像docker文件：

```
        from anolisos

        MAINTAINER ZJY

        RUN yum install -y epel-release

        RUN yum install -y openssh-server passwd nmap-ncat net-tools java-21-openjdk-devel
        supervisor

        RUN \cp -f /usr/share/zoneinfo/Asia/Shanghai /etc/localtime && echo -e 'ZONE="Asia/
        Shanghai" \nUTC=false \nARC=false' > /etc/sysconfig/clock && echo_supervisord_conf > /etc/
        supervisord.conf && echo -e "\n[program:sshd]\ncommand=/usr/sbin/sshd -D\n" >> /etc/
        supervisord.conf && sed -i "s/nodaemon=false/nodaemon=true/g" /etc/supervisord.conf &&
        ssh-keygen -q -t rsa -N "" -f /etc/ssh/ssh_host_rsa_key && ssh-keygen -q -t ecdsa -N ""
        -f /etc/ssh/ssh_host_ecdsa_key && ssh-keygen -q -t ed25519 -N "" -f /etc/ssh/
        ssh_host_ed25519_key && echo "linux" | passwd --stdin root

        ENV LANG zh_CN.UTF-8
        ENV LANGUAGE zh_CN:zh
        ENV LC_ALL zh_CN.UTF-8
```

CMD ["/usr/bin/supervisord", "-c", "/etc/supervisord.conf"]

制作基础镜像命令：

docker build -f os_base.docker -t os_base .

制作后端服务镜像docker文件：

FROM os_base

MAINTAINER ZJY

COPY zpaas-lowcode-starter-0.0.1-SNAPSHOT.jar /zpaas-lowcode-starter-0.0.1-SNAPSHOT.jar

RUN mkdir /opt/zpaas-lowcode && mv /zpaas-lowcode-starter-0.0.1-SNAPSHOT.jar /opt/zpaas-lowcode && echo -e "\n[program:zpaas-lowcode]\ncommand=java -jar -server /opt/zpaas-lowcode/zpaas-lowcode-starter-0.0.1-SNAPSHOT.jar \ndirectory=/opt/zpaas-lowcode \n" >> /etc/supervisord.conf

CMD ["/usr/bin/supervisord", "-c", "/etc/supervisord.conf"]

制作后端服务镜像命令：

sudo docker build -f zpaas-lowcode.docker -t zpaas-lowcode .

后端服务启动命令：

sudo docker run -itd -p 18043:18043 zpaas-lowcode

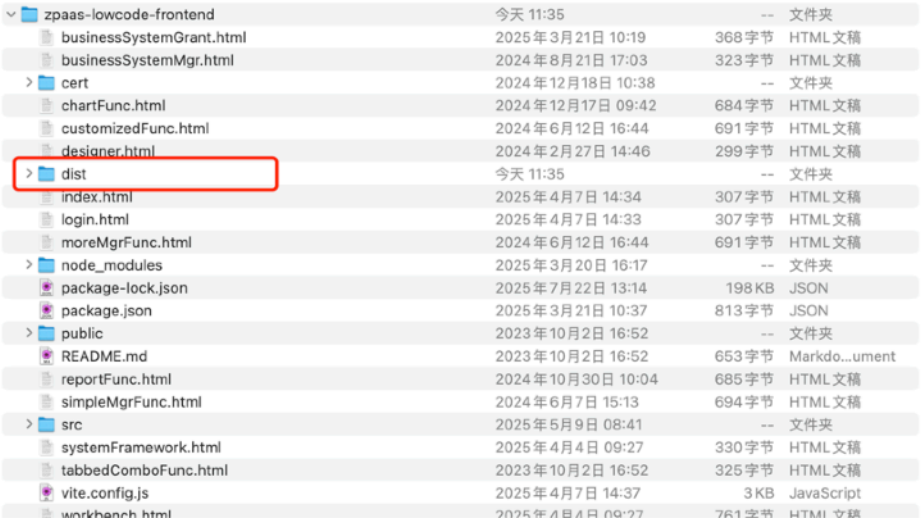
4. 前端服务部署

（1）切换到前端服务工程zpaas-lowcode-frontend并进行编译打包

cd zpaas-lowcode-frontend

npm run build

（2）在前端工程下会生成dist目录



（3）在服务器上安装nginx

yum install nginx

（4）将dist目录下的文件上传到服务器，并复制到nginx的目录下/usr/share/nginx/html/

（5）修改nginx的配置文件，示例如下：

```
worker_processes auto;
error_log  error.log;
#error_log  logs/error.log  notice;
#error_log  logs/error.log  info;

#pid        logs/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       mime.types;
    default_type  application/octet-stream;

    #log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
    #                  '$status $body_bytes_sent "$http_referer" '
    #                  '"$http_user_agent" "$http_x_forwarded_for"';

    #access_log  logs/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    #keepalive_timeout  0;
    keepalive_timeout  65;

    gzip  on;

    client_max_body_size   20m;
    server {
        listen       11443;
        server_name  zpaas-lowcode-front;

        ssl on;
        #ssl证书的pem文件路径
        ssl_certificate  /cert/ca.cer;
        #ssl证书的key文件路径
        ssl_certificate_key /cert/ca-key.pem;
        #charset koi8-r;

        #access_log  logs/host.access.log  main;
        location / {            #前端服务的配置
            root   /usr/share/nginx/html;
            index  index.html index.htm;
            try_files $uri $uri/ /index.html;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        }
```

```
        location /lcdp-proxy/ {        #后端服务的转发配置
            proxy_pass https://192.168.0.100:18043/;  #IP地址及端口需要相应调整
            proxy_cookie_path /lowcode /lcdp-proxy/lowcode;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        }

        #error_page  404              /404.html;

        # redirect server error pages to the static page /50x.html
        #
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   html;
        }
    }
}
```

（6）重启nginx服务

（7）服务的访问地址示例如下：
https://129.153.118.144:11443

内置的用户：
demo/adl28m2@ma(a!*D2
demo1/adl28m2@ma(a!*D2
admin/adl28m2@wre5(a!*D2

（8）如果需要采用docker方式部署，可以参考以下脚本
前端服务镜像制作docker文件：

```
FROM nginx
MAINTAINER ZJY
COPY dist/  /usr/share/nginx/html/
COPY nginx.conf /etc/nginx/nginx.conf
RUN mkdir /cert && COPY cert/ /cert/
RUN echo 'echo init ok!!'
```

前端服务镜像制作命令：

```
sudo docker build -f zpaas-lowcode-front.docker -t zpaas-lowcode-front .
```

前端服务启动命令：

```
sudo docker run -itd  -p 11443:11443  zpaas-lowcode-front
```

5. 其他

低代码平台的源码已经提交到GitHub：https://github.com/zjyzju/zPaaS-lowcode，后续将通过更多的文章，逐步对低代码平台的设计理念/思想、具体的设计、部署以及操作说明进行细化，敬请关注。

另外在一台云服务器上部署了一个演示环境，有兴趣的小伙伴可以通过演示环境进行试用：

访问地址：https://129.153.118.144:11443

试用账户： demo/adl28m2@ma(a!*D2

demo1/adl28m2@ma(a!*D2

注：（1）该演示环境部署在一台免费的云服务器上，且位于国外，在国内访问响应会比较慢

（2）系统管理子系统的菜单框架引用的@micro-zoe/micro-app@1.0.0-rc.24组件，该组件在不同浏览器中存在兼容性问题，苹果的safari浏览器访问一切正常，但是windows的Edge和谷可的Chrome会存在页面刷不出来的情况，需要多等一会儿（有解决方案的可以邮件发给我，多谢！https://github.com/jd-opensource/micro-app/issues/1559#issuecomment-2784997902）。

如果有低代码平台以及AI相关的讨论，可以发送邮件到我的邮箱：zjyzju@163.com 。